Traffic Sign Recognition using CNN

Raghavendra Reddy
Artificial Intelligence and Machine Learning Program
Lambton College
Toronto, Ontario, Canada
C0851724@mylambton.ca

Shravan Kumar Reddy
Artificial Intelligence and Machine Learning Program
Lambton College
Toronto, Ontario, Canada
C0833124@mylambton.ca

Jebin George
Artificial Intelligence and Machine Learning Program
Lambton College
Toronto, Ontario, Canada
C0850509@mylambton.ca

Nikeshh Vijayabaskaran
Artificial Intelligence and Machine Learning Program
Lambton College
Toronto, Ontario, Canada
C0849544@mylambton.ca

Abstract* - There is a growing demand for a reliable traffic sign recognition system that can quickly and accurately identify traffic signs to ensure safety as autonomous driving vehicles gain popularity throughout the world. We create a CNN that can categorize 43 different traffic signs from the benchmark dataset for German Traffic Sign Recognition. The importance of the traffic signs we see on the roadways in our daily lives as drivers cannot be overstated. They offer the users of the road's vital information. As a result, they will need to gradually control their driving behavior and make sure they strictly adhere to the traffic laws that are in effect at the time without endangering other motorists or pedestrians. Traffic sign classification is used to identify and categorize traffic signs in order to alert and warn a driver in advance to prevent breaking the law. The suggested approach substantially addresses some drawbacks of the current categorization systems, such as inaccurate predictions, hardware cost, and maintenance. The suggested method employs a convolutional neural network to create a traffic sign classification algorithm. It also includes the capability of traffic sign web cam detection. As a result, the driver will be able to see the sign right in front of his or her eyes on the display screen, saving time from having to manually check the traffic sign each time.

Keywords— Image Classification, CNN

I. INTRODUCTION

Advanced Driving Assistance Systems (ADAS) technology is constantly evolving. Object detection is crucial to the further advancement of Intelligent Driving and Traffic Safety in the approaching trend of self-driving vehicles. It may be a matter of life and death for autonomous vehicles to be able to recognize, classify, and respond to various traffic signs. This means that the device must be precise and quick to identify immediately. Nowadays, machine learning algorithms are more important than ever. Among the applications of machine learning, spam filtering, speech interpretation, facial recognition, and road sign identification are just a few. Traffic sign categorization and recognition software can be used in high-traffic areas to automatically recognize traffic signs. As soon as a traffic sign is identified, the system instantly displays the sign's name. Therefore, even if the driver misses a sign or has a moment of distraction, it will still be picked up. This aids in warning the drivers appropriately and prohibiting certain behaviors like excessive speeding. Additionally, it relieves the driver's burden, improving comfort. Hence, making sure to pay attention to and follow the traffic signs as necessary.

II. MOTIVE OF PROJECT

One of the key reasons for choosing this project is to design and improve such a system because it has the potential to save lives and money. The major goal of our effort is to design such a system using deep learning methods.

III. METHODOLOGY

A. Data Collection

For the classification module, the German Traffic Sign dataset will be used. This dataset consists of about 51822 images which is further classified into 43 classes. The data contains testing and training image data and image information in CSV files. The reason for this dataset is because

1. It consists of large number of images
2. The traffic signs are of different variety, background, and colour variation which in turn will help the model to perform accurately.

B. Data Analysis

For the dataset we explored how many files present in each class so that we wanted to know whether the data is biased or good to conduct an experiment. And we also analyzed the images since the quality of images should be good in order to train the model good. The below figure1 represents Number of images in each class. The below figure2 represents the quality of images and variety of images belongs to different classes.

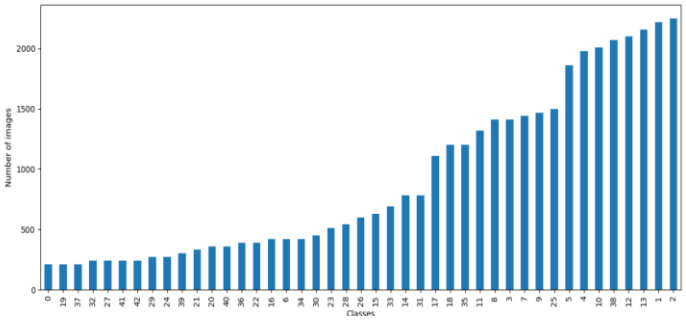

Figure1: Class Vs No of Images

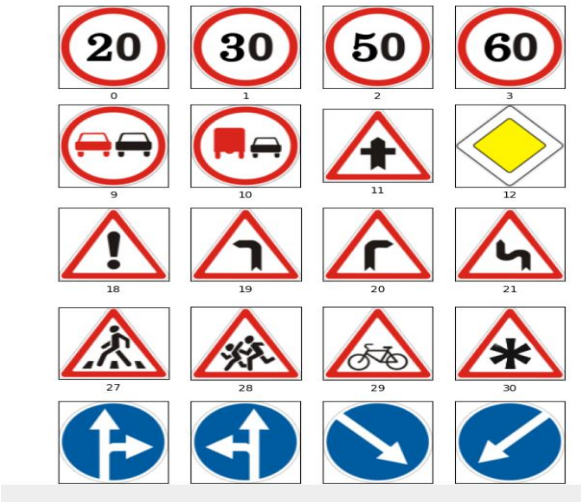

Figure2: Data Analysis

C. *Data PreProcessing*

This an image dataset, so we checked any errors while loading an image from directories. We reshaped the images into one standard size, so that model can get trained faster. The below figure 3 represents the data pre-processing steps

```
    try:
        #Open image
        image = Image.open(path + '/' + a)
        #(Image)Resizes to 30x30
        image = image.resize((30, 30))
        #Conversion of image into an array
        image = np.array(image)
        # Append the image to "data" list
        list_data.append(image)
        # Append the label to list
        list_labels.append(i)
    #error message
    except:
        print('Error loading images!')
```

Figure3: Data preprocessing

D. *Modelling*

Here we are building an CNN Model for image feature extraction and classification. We also bringing the contrast of two models that is A normal baseline CNN model with Pre-trained model in order to understand the behavior and factors impacting the model. The below figure 4 represents the model architecture of CNN.

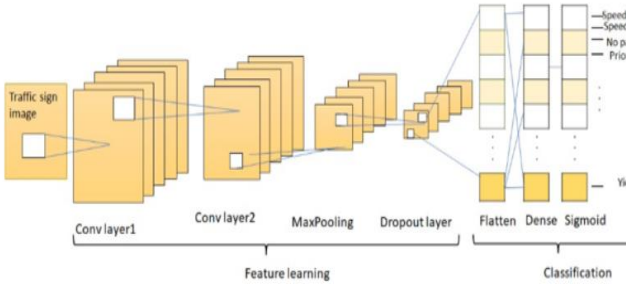

Figure4: Model Architecture

I. A Convolutional Neural Network CNN was built from the scratch for the purpose of feature extraction and classification. For the model we have used Convolution2D, Max pooling layer, ReLu Activation function, flattening layer, Dropout in order to avoid overfitting the model. The below figure 5 represents an entire summary of model that have been built.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	2432
conv2d_1 (Conv2D)	(None, 22, 22, 32)	25632
max_pooling2d (MaxPooling2D)	(None, 11, 11, 32)	0
dropout (Dropout)	(None, 11, 11, 32)	0
conv2d_2 (Conv2D)	(None, 9, 9, 64)	18496
conv2d_3 (Conv2D)	(None, 7, 7, 64)	36928
max_pooling2d_1 (MaxPooling2D)	(None, 3, 3, 64)	0
dropout_1 (Dropout)	(None, 3, 3, 64)	0
flatten (Flatten)	(None, 576)	0
dropout_2 (Dropout)	(None, 576)	0
dense (Dense)	(None, 43)	24811

Figure5: A Simple CNN Model Summary

II. We also used pre-trained model such as VGG model which is trained on more than million images from the image net database. One of the best vision model designs to date is the convolution neural network (CNN) architecture GG16. VGG16 employs convolution layers with a 3x3 filter, a stride 1, and a maxpool layer of a 2x2 filter, stride 2, as opposed to having a lot of hyper-parameters. Throughout the entire architecture, convolution and max pool layers are arranged in the same manner. It features two fully interconnected layers at the very end, followed by a SoftMax for output. The 16 in VGG16 denotes the fact that there are 16 layers with weights. This network has 138 million (approximately) parameters, making it a fairly sizable network. The below figure 6 represents an entire summary of model that have been built.

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 1, 1, 512)	14714688
batch_normalization (BatchNormalization)	(None, 1, 1, 512)	2048
flatten (Flatten)	(None, 512)	0
dropout (Dropout)	(None, 512)	0
dense (Dense)	(None, 1024)	525312
dense_1 (Dense)	(None, 43)	44075

Total params: 15,286,123
Trainable params: 15,285,099
Non-trainable params: 1,024

Figure6: VGG16 Model Summary

E. *Train, Test and Validation*

We split the dataset into training set and testing set into 80% and 20% respectively. We also include additional data with validation data in order to analyze the performance of model with real time images.

F. *Data Visulaization*

We visualized the number of images in each class and plotted using box plot. And also, the model results in order to understand which model performs better and why?

The below figure 7 represents the plot for accuracy and loss for train and validation dataset respectively for simple CNN Model. We can observe from the figure 7 the data was trained for 15 epochs with batch size 64 and we have used categorical cross entropy loss function in order to calculate the loss. As number of epochs increases we can observe the accuracy of model was increasing steadily and loss was minimizing so that model can predict the most appropriate class of that image belongs. We also can see there is less variance between train and validation so the chances of model being overfit is very less. We also used dropout layer in order to avoid overfitting of the model.

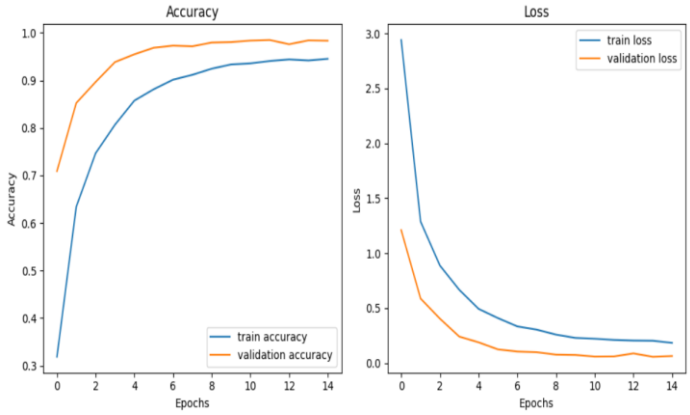

Figure7: Simple CNN Accuracy & Loss Plot

The below figure 8 represents the plot for accuracy and loss for train and validation dataset respectively for VGG16 Model. We can observe from the figure 8 the data was trained for 2 epochs with batch size 64 and we have used categorical cross entropy loss function in order to calculate the loss. As number of epochs increases we can observe the accuracy of model was increasing steadily and loss was minimizing so that model can predict the most appropriate class of that image belongs. We also can see there is less variance between train and validation so the chances of model being overfit is very less. We also used dropout layer in order to avoid overfitting of the model.

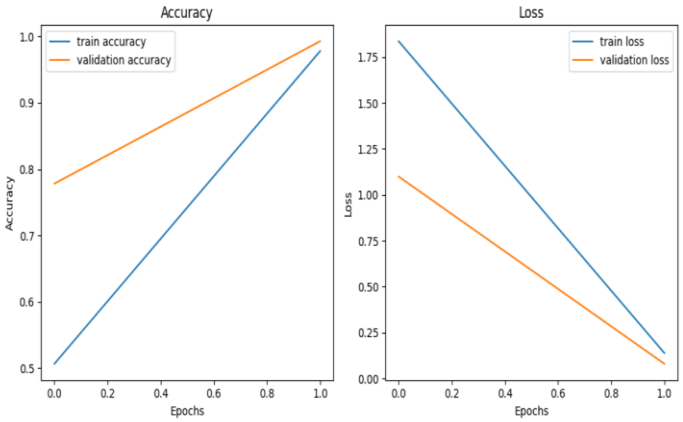

Figure8: VGG Accuracy & Loss Plot

G. *Model Evaluation*

We can assess a machine learning or deep learning model using measures such as accuracy, loss, precision, and so on. The metrics differ depending on the type of algorithm, but in this instance, we'll use accuracy to gauge how well the model is working. To determine this accuracy, we are using the photographs from the testing folder.

The below figure 9 represents the accuracy score of predicted classes vs the labelled classes for simple CNN model. We got around 95.305% accuracy for the model.

```
In [34]: # Evaluate model
         print('ACCURACY: {} %'.format(round(accuracy_score(labels, classes_x) * 100, 3)))

         ACCURACY: 95.305 %
```

Figure9: CNN Model Evaluation

The below figure 10 represents the accuracy score of predicted classes vs the labelled classes for VGG16 model. We got around 95.273% accuracy for the model.

```
In [23]: # Evaluate model
         print('ACCURACY: {} %'.format(round(accuracy_score(labels, classes_x) * 100, 3)))

         ACCURACY: 95.273 %
```

Figure10: VGG16 Model Evaluation

IV. UI Design

We also developed a simple UI using ReactJS and Flask App. We generated a http request from flask and verified the API in postman and then we exposed a port using ngrok. Then we integrated with ReactJs Script. The below figure 11 demonstrates that we can upload any traffic sign related image and click on predict which in turn will generate a response of what that particular traffic sign describes.

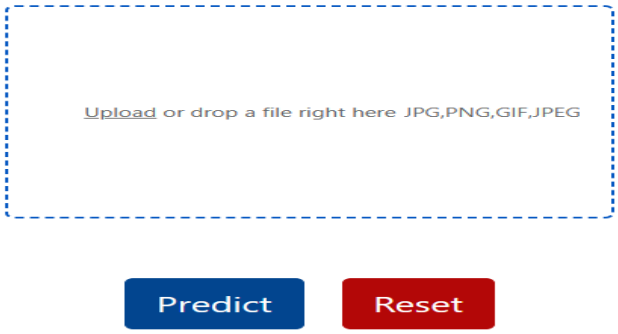

Figure11: UI for Traffic Sign Classification

V. Results

Finally, the model is integrated with UI. We performed by providing a sign and the model was able to detect which class it belongs to and gives a signal notification or label text. The below figure 12 represents the output of the model which is predicted the traffic sign which we have uploaded

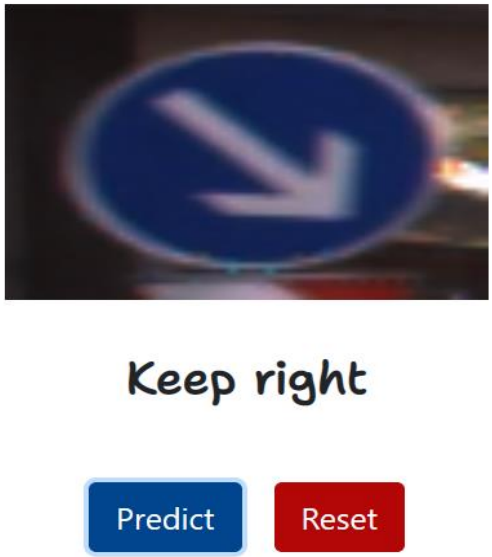

Figure12: Final End Result

VI. Future Scope

All motorists who are operating a motor vehicle on the road can benefit from traffic signs. Traffic signs direct drivers to obey all traffic laws and to not obstruct pedestrians in any way. The detection and subsequently the classification may be impacted by environmental factors such as motion blur, vehicle vibration, air pollution, air quality, illumination, shadow, and distance (the sign is pretty far away). Therefore, to address these problems, more study and developments are required. Additionally, it's possible that some traffic indicators can't be precisely predicted. Techniques like one-hot encoding and augmentation can be utilized for this. Image augmentation entails repositioning, enlarging, and rotating the images (if applicable). This device aids the driver in viewing the sign right in front of their eyes on the screen. This saves the time and effort of manually examining each traffic sign board to see if any are present, figuring out what kind of sign it is, and acting appropriately. Building smarter cars, such as those with autonomous driving, where the system automatically detects, recognizes, and displays a traffic sign, has a wide use for traffic sign classification.

VII. Conclusion

A CNN was successfully built for traffic sign classification and compared with pre-trained model. In future scope, this model can be used in self autonomous drive vehicles. The suggested system is straightforward and performs classification fairly accurately on both the GTSRB dataset. Finally, the model can successfully capture images and predict them accurately even if the background of the image is not very clear. Convolutional Neural Networks (CNN) are used in the proposed system to train the model. The final accuracy we got is around 95% on the test dataset using both CNN and VGG16 Model. The algorithm also does quick and precise web cam predictions that are quite accurate. The advantages of the "Traffic Sign classification and detection system" are often geared towards driver comfort. There are disadvantages to traffic sign classification in spite of its benefits. There may be instances when the traffic signs are obscured or difficult to see. This can be risky because the driver won't be able to control how fast his car is going. This could result in accidents that put other drivers and pedestrians in danger, necessitating more study.

VIII. References

[1]. Akatsuka, H., & Imai, S. (1987). Road signposts recognition system (No. 870239). SAE Technical Paper.

[2]. Shao, F., Wang, X., Meng, F., Rui, T., Wang, D., & Tang, J. (2018). Real-time traffic sign detection and recognition method based on simplified Gabor wavelets and CNNs. Sensors, 18(10), 3192.

[3]. Li W., Li D., & Zeng S. (2019, November). Traffic Sign Recognition with a small convolutional neural network. In IOP conference series: Materials science and engineering (Vol. 688, No. 4, p. 044034). IOP Publishing

[4]. Alghmgham, D. A., Latif, G., Alghazo, J., & Alzubaidi, L. (2019). Autonomous traffic sign (ATSR) detection and recognition using deep CNN. Procedia Computer Science, 163, 266-274

[5]. Rusanov, I. S., 2020. Traffic Sign Recognition Problem with GTSRB, Python and Keras. [Online] Available at: https://medium.com/ai-in-plain-english/traffic-sign-recognition-problem-with-gtsrb-python-and-keras-21782b7b3520