**AI PROJECT REPORT**

**TRAFFIC SIGN CLASSIFIER**

***by***

**20K-0441 MUHAMMAD FAHAD| BCS-6C**

**20K-0278 MUHAMMAD AYYAN TAHIR | BCS-6A**

**20K-0278 SYED ABDULLAH | BCS-6A**

**Contents:**

**1.Abstract**

**2.Literature Review**

**3.Introduction and Motivation**

**4.Methodology**

**5.Results**

**6.Conclusion**

**Abstract:**

A CNN based web application that will classify the image provided using the classification model trained on a dataset using numerous images of various traffic signs categorized on the basis of their names. The dataset used is the German Traffic Sign Benchmark dataset and it has more than 40 classes, more than 50,000 images in total, large, lifelike database. After the CNN model is trained, it is integrated into a web application, providing a user-friendly interface for traffic sign classification. Users can upload images of traffic signs to the application, which then passes the image through the pre-trained CNN model for classification. The model's classification result is displayed to the user, providing immediate feedback on the identified traffic sign.

**Introduction and Motivation:**

Traffic sign recognition is essential for road safety and efficient traffic management. This project aims to develop a CNN-based web application for traffic sign classification, utilizing the German Traffic Sign Benchmark dataset. With over 50,000 images categorized into more than 40 classes, the dataset provides a comprehensive representation of various traffic signs.

Traditional methods for traffic sign recognition have limitations in handling real-world variations. CNNs offer a solution by automatically learning and extracting relevant features from images, improving accuracy and robustness. The German Traffic Sign Benchmark dataset enables the training of a model capable of handling real-world scenarios effectively.

The developed web application provides a user-friendly interface, enabling drivers and traffic authorities to classify traffic signs easily. Real-time feedback enhances decision-making and road safety. This project's outcomes have implications for advanced driver assistance systems, autonomous vehicles, and traffic management, improving performance and safety.

The goal of building a CNN-based traffic sign classification system is to improve road safety by increasing the accuracy and reliability of traffic sign recognition. Doing so will help in reducing the chance of accidents caused by human error.

**Literature Review:**

Traffic sign classification is a crucial task in intelligent transportation systems, aiding in road safety, autonomous driving, and traffic management. Convolutional Neural Networks (CNNs) have proven to be powerful techniques for image recognition and classification. This literature review provides an overview of recent advancements in traffic sign classification using CNNs, covering methodologies, datasets, and performance evaluation metrics.

Traffic sign classification is challenging due to variations in illumination, occlusion, and viewpoint. CNNs have achieved remarkable success in image classification tasks and are widely used for traffic sign recognition. Different CNN architectures, such as LeNet-5, AlexNet, GoogLeNet, VGGNet, and ResNet, are discussed, highlighting their strengths and weaknesses in traffic sign recognition. Preprocessing techniques play a vital role in improving CNN model performance. Techniques such as image resizing, normalization, histogram equalization, and data augmentation are examined, along with their impact on CNN-based traffic sign classifiers.

Several benchmark datasets, including the German Traffic Sign Recognition Benchmark (GTSRB), LISA Traffic Sign Dataset, and Belgian Traffic Sign Dataset, are commonly used for training and evaluating traffic sign classification models. The characteristics, size, and annotation details of these datasets are discussed.Training strategies, including loss functions, learning rate schedules, weight initialization, and transfer learning, are reviewed. Fine-tuning strategies, such as freezing and unfreezing layers, are also explored to enhance model performance. Performance evaluation metrics such as accuracy, precision, recall, and F1-score are commonly used to assess traffic sign classification models. Their applications in the field are discussed. A comparative analysis summarizes the performance of various CNN models and techniques discussed in the review, highlighting key similarities, differences, and trends observed in the literature.

In conclusion, CNNs have proven effective in achieving high accuracy in traffic sign classification. The review identifies potential research directionand open challenges that need to be addressed to further improve the performance and practicality of traffic sign classification systems.

**Methodology:**

**Dataset:**

A 600mb dataset of more than 50000 images was present on kaggle . The dataset we have segregated in categorically on the basis of meaning of that traffic signs, but these segregated data was labeled numerically where each number is having it’s own meaning.We split the datas

**Image Preprocessing:**

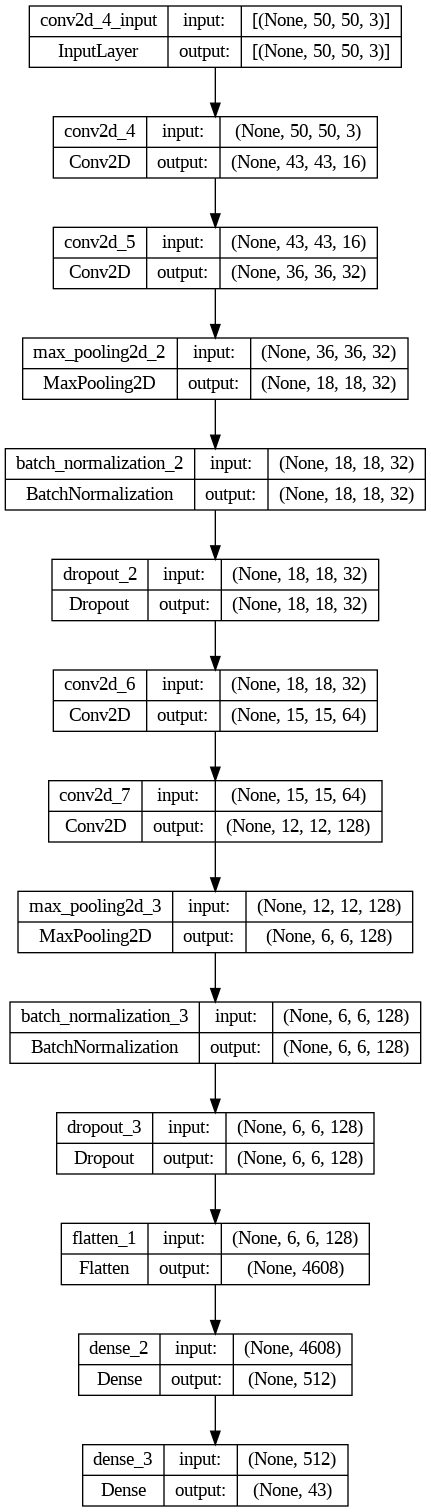
During preprocessing, the images undergo resizing to achieve uniform dimensions,50\*50\*3 to be exact, facilitating effective model training. Next, the images are converted into a numerical representation, enabling the model to process pixel values and learn patterns. The transformed images maintain their relationships with the respective traffic sign classes.

**Model:**

We picked a pre-trained sequential CNN model which had 95% accuracy however we decided to fine-tune the model by using different kernel sizes and no of filters at each convolution and added some additional layers and batch normalization as well.After every two convolutional layers Max pooling was applied with a stride of 2. Dropouts were used to cater the overfitting problem and ReLu activation function was used to address the vanishing gradients .We then re-trained the model and managed to improve the testing accuracy to 97%.The model architecture can be seen in Figure 1 below.

**Training:**

We trained the new model over the Train folder in the dataset that had 38000+ images with 2.5Million parameters by using the Gpu runtime on Google colab.We had a 75:25 testing to validation split and we ran 20 epochs with the batch size set to 32 which meant that each epoch contained over 981 samples and the learning rate was set to 0.001.We achieved amost 98 percent validation and testing accuracy Figure 2 and Figure 3 show the Accuracy and Loss curves below



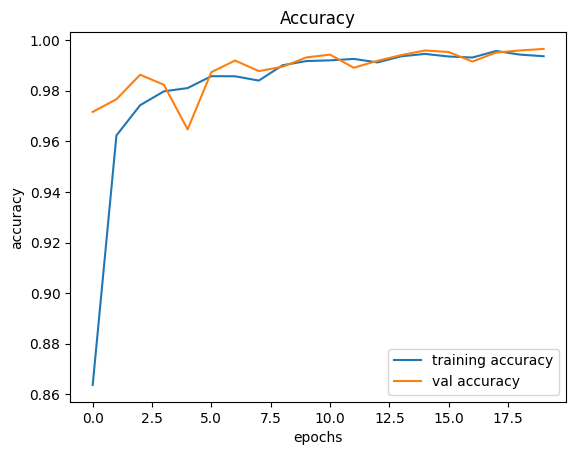


Figure 2.

Figure 1.

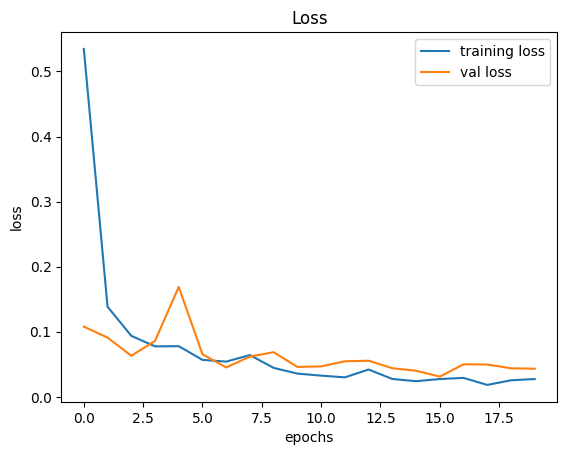


Figure 3.

**Testing:**

The trained model was evaluated on a separate test dataset to assess its accuracy. The performance of the model was evaluated using the sklearn accuracy score metric which provided an indication of how well the model was performing in classifying the images. We achieved an accuracy of score of 97% and plotted a confusion matrix heatmap as well to see how correct the predictions on the diagonal are.

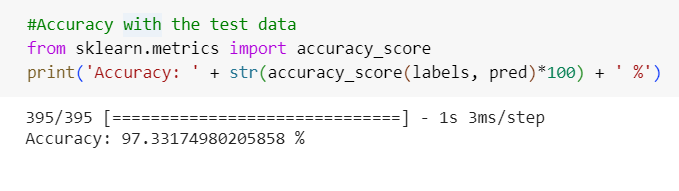
****

Figure 4.

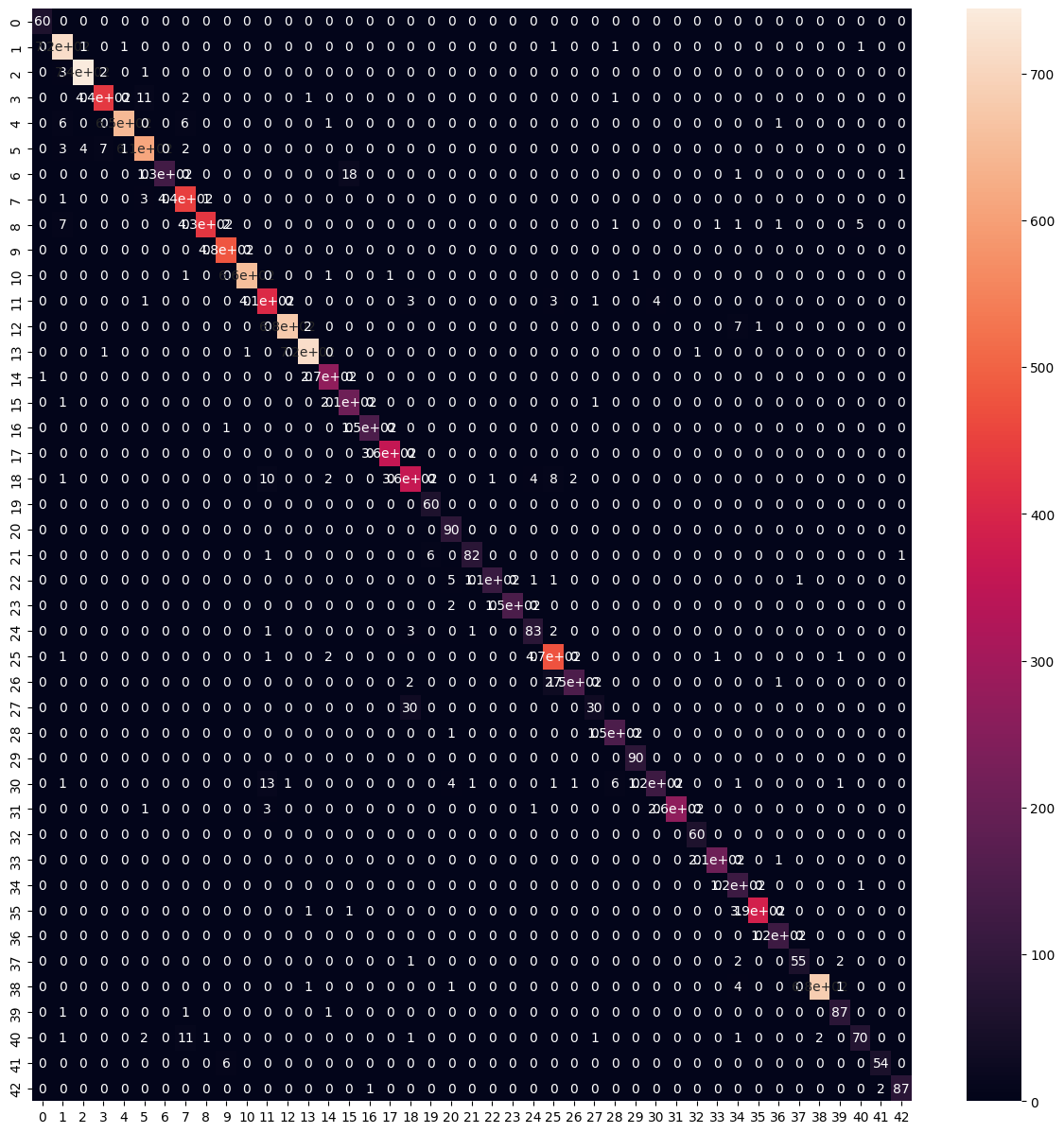


Figure 5.

**Web-app:**

The model was trained and saved and then loaded into the web-app designed using the gradio framework in python . The web has a modern and an easy to use interface and it uses the trained model saved as “Dlpproj.h5” at the backend to classify the images input by the users. The web app is also deployed on the server and anyone can access it using a live link generated through gradio.

**Results:**

After training, we evaluated each model on a test folder in the dataset that had 12000+ images , we passed the 5 random images at a time from the test set and then predicted the signs. The design function generated a visual representation of five images with their respective actual and predicted labels, highlighting any discrepancies between the two labels using different text colors.





Figure 6.

We also created a classify\_image() function to take an input image on the console and predict its label and the corresponding sign



Figure 7.

The web app designed using the gradio framework also goes live and anyone can access it using the generated link. The easy to use interface is shown in the figures below.



The user can upload or drag and drop a traffic sign image here to identify it by clicking on the submit button. Few examples are shown below

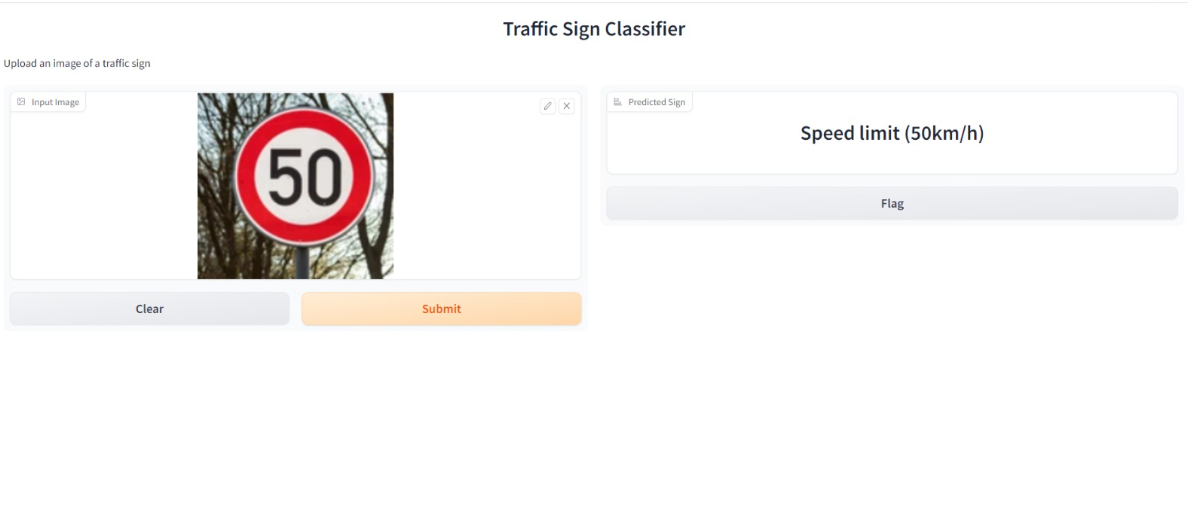


Figure 7.



Figure 8.

It can be seen that it correctly predicts the “50km/h speed limit” and the “General Caution” signs present in the input images above

**Conclusion and Future Work:**

The model achieved promising results, demonstrating its ability to effectively classify traffic signs with a high degree of accuracy. By training the model on a large dataset and using advanced deep learning algorithms, we were able to achieve significant improvements in classification performance. The traffic sign classifier web app is capable of accurately classifying traffic signs, with the potential for practical applications in areas such as autonomous vehicles, driver assistance systems, and traffic control. Further improvements and fine-tuning can be explored to enhance the model's performance and address any remaining challenges in real-world scenarios. In order to further enhance this project, the system can be extended to accommodate real-time data input by utilizing a camera to capture live images. This addition would allow the classifier to process and classify traffic signs in real-time, providing instant feedback and analysis.

**References:**

Yacharki. (n.d.). Traffic Signs Image Classification - 95.22% CNN. Retrieved from <https://www.kaggle.com/code/yacharki/traffic-signs-image-classification-95-22-cnn>

[1] Nain, N. (2021, December 9). Traffic Signs Recognition Using CNN and Keras in Python. Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2021/12/traffic-signs-recognition-using-cnn-and-keras-in-python/>.