Traffic Sign Recognition

Group Members
Jumana Lightwala: C0892358
Niral Shah: C0894470
Shivamkumar Patel: C0907758
Shreyash Banawala: C0891309

Introduction

Greetings and welcome to the Traffic Sign Recognition Web App project introduction. In this session, we'll explore an intriguing application that uses cutting-edge technology to transform how we recognize and c lassify traffic signs.

By developing a robust model capable of accurately detecting various t ypes of traffic signs, we seek to mitigate risks on roads and improve traffic management systems.

This project holds significant potential to revolutionize transportati on by providing advanced solutions for real-time sign recognition and autonomous vehicle navigation.

Waterfall Methodology

Deployment and
Maintenance
System Design
Testing
Implementation
Requirements Gathering
03
04
02
05
01

The waterfall methodology is used in our Traffic Sign Detection System project, which breaks work into phases such as requirements, design, implementation, testing, deployment, and maintenance. We effectively m anage resources when we have well-defined goals in advance.
The waterfall approach offers a strong foundation for our CNN-based so lution by emphasising planning and progressive development.

Key Points:
Recognition of Traffic Signs: Traffic signs are essential for maintain ing road safety and traffic control. On the other hand, manual recogni tion can be laborious and error-prone.
Automated Solution: To meet this difficulty, our research creates an a utomated solution that uses machine learning methods to identify traff ic sign labels from pictures.
Practical Importance: Correct identification of traffic signs leads to safer roadways, fewer collisions, and an improved driving experience.

Problem Statement

Solution Overview

Introduction to the Web App: Traffic Sign Identification The purpose o f the web application is to offer an interactive platform for machine learning-based traffic sign recognition and classification.
Key Features: The app's main features include the ability to upload im ages, real-time prediction, instructional materials, and insights into the training of models.
Educational Value: By learning about effective training methods, pictu re preprocessing, and convolutional neural networks (CNNs), users can better comprehend machine learning ideas.

How It Works

User Uploads Image: Using the web app's easy-to-use interface, start b
y uploading a picture of a traffic sign.

Preprocessing for Consistency: To guarantee consistent input for the m
odel, preprocessing is applied to the supplied image. This includes sc
aling and color channel management.

Model Prediction: For prediction, the image is sent into a Convolution
al Neural Network (CNN) that has been trained.

Quick Results: The user can see the traffic sign's anticipated label o
n the web app in a matter of seconds.

Efficient Image Handling

Model Architecture

CNN Architecture: Specifically created for image identification applic
ations, the CNN model is a deep learning architecture.

Key Layers: Convolutional, pooling, and thick layers make up the model.

Feature Extraction: To extract features from input images, such as edg
es and forms, convolutional layers utilize filters.

Pattern Recognition: As the network goes through layers, patterns are
learned, allowing the model to recognize different aspects of traffic
signs.

Reading the images

This function reads the all images at first. In this step first, image
s converted into 30x30 size from train dataset.

Then all resized image list converted into numpy arrays.

Splitting the training and test datasets by using train_test_split() function.

Converting the labels into one hot encoding.

Trained data sent to model.

Model Understanding

Model Understanding

Model Training Techniques

Optimization Strategies:
The Adam optimizer is used to improve the training of our model. Adam customizes learning rates for each parameter, which leads to more rapi d convergence and effective updates.
Regularization:
We use a regularization method called dropout to avoid overfitting. In order to encourage a more comprehensive model during training, dropou t entails randomly deactivating a portion of neurons.
Dropout Layer Implementation:
The "Dropout" layers in the model architecture purposefully deactivate neurons at a rate of 0.25, which forces the network to use different paths in order to learn information.
Model Training
After training the model this function is used to save the training mo del as training takes around 45 mins and lots of GPU and memory. We sa ved the trained model and use it wisely to predict the traffic signs.

The commented code is used to save model.

Training Insights

Visual Representations: During model training, graphs show trends in a

ccuracy and loss over epochs.
Accuracy Curve: This shows how the model's prediction accuracy increases as it is trained.
Loss Curve: As the model gains knowledge from training data, the loss
curve shows how its error falls.

Front-end Technology
There are vast number of technologies available for front-end developm
ent like Angular 13, React Js, Vue Js, Bootstrap, Flask, etc.

Why Flask?
For this project, We decided to go with flask because of several reasons mentioned below:

Flask is a lightweight and flexible web framework for Python, making i
t ideal for building web applications quickly and efficiently.

It provides developers with the tools to create web applications in a
simple and elegant manner, allowing for rapid prototyping and developm
ent.

Flask follows the WSGI (Web Server Gateway Interface) specification an
d is often used with Jinja2 templating engine for dynamic HTML generat
ion.

■As aforementioned points, We decided to go with the flask and the mai
n advantage for our project is There will not to make backend it direc
tly deal with the models in .py file.
Front End Insights
Front End Insights

Predicting Traffic Signs

Seamless Prediction: A traffic sign's associated label is predicted by the "predict_model" function, which accepts an image as input.
Alpha Channel Handling: Images containing transparency information, or alpha channels, are processed to preserve the RGB information while e liminating the alpha channel.
Resizing: In order to ensure compatibility with the model's input size , images are scaled to a consistent dimension before to prediction.

Future Enhancements

Innovation Roadmap: We're dedicated to ongoing development, working to increase the capabilities and experience of users.
Real-Time Capture: In the future, users may be able to utilize the cam era on their device to predict traffic signs in real-time.
Batch Prediction: Simplified features can enable users to upload and f orecast several pictures at once.

Conclusion

Finally, the Traffic Sign Recognition Web App combines safety and tech nology by precisely interpreting signs to encourage cautious driving.
Our experience has helped to demystify machine learning and encourage research beyond forecasts.

Finally, let's not forget that the software serves as an educational p ortal into the field of artificial intelligence in addition to being a tool. Come celebrate with us this milestone and the app's promise to make the future safer and more informed.

● Kaggle for traffic signs classification with CNN - https://www.kaggl e.com/code/valentynsichkar/traffic-signs-classification-with-cnn

● TensorFlow. (n.d.). TensorFlow Documentation. Retrieved from https://www.tensorflow.org/
● PyTorch. (n.d.). PyTorch Documentation. Retrieved from https://pytorch.org/docs/stable/index.html
● Sklearn. (n.d.). Scikit-learn Documentation. Retrieved from https://scikit-learn.org/stable/documentation.html
● Github link: https://github.com/shreyash5965/trafficsign

References