

Final Project

Enhanced Traffic Sign Recognition System

Kumara Swamy Padigeri (U45008503)
Muma College of Business
Tampa, Florida
pk24@usf.edu

Sunil Yanagandula (U82205360)
Muma College of Business
Tampa, Florida
sunily@usf.edu

Abstract

This project offers an innovative traffic sign identification system that greatly improves accuracy and robustness in a variety of environmental situations by utilizing autoencoders and convolutional neural networks (CNNs). Our solution is designed to improve the dependability of driver-assistance and autonomous cars by efficiently handling obstacles including varying lighting, different types of weather, and occlusions.

I. INTRODUCTION

Accurately recognizing traffic signs is crucial for the functioning and safety of the rapidly developing autonomous and assisted driving technologies, as well as for their general adoption and incorporation into everyday transportation. The identification of traffic signs is a vital component that connects the sensory input gathered by car sensors with the useful information required for these systems to make navigational decisions in real time. Even though they work well in normal circumstances, today's technologies frequently break down in unusual environments like dim lighting, bad weather, or physical obstructions. These flaws put vehicle safety and operational dependability at serious risk by causing mistakes in sign interpretation.

This project proposes a dual-model architecture traffic sign identification system that combines the powers of convolutional neural networks (CNNs) for pattern recognition with the dimensional reduction capabilities of autoencoders to handle these difficulties. This combination is intended to improve the system's robustness by strengthening its capacity to recognize and classify traffic signs correctly in a variety of challenging environmental settings. By encoding the input images into a simplified representation and eliminating noise and extraneous features, autoencoders aid in sustaining performance under less-than-ideal circumstances. The CNNs, which are well-known for their effectiveness in image classification tasks, use these refined features to accurately classify traffic signs.

The development and implementation of this hybrid model are described in this study, along with the methods used for data preparation, model training, and performance evaluation. The objective of this project is to create safer and more dependable autonomous driving systems that can function well in a variety of operating circumstances by strengthening the recognition system's robustness to environmental perturbations. The details of the model architecture, training protocols, data preparation methods, and the all-encompassing outcomes of this novel methodology will be covered in detail in the parts that follow.

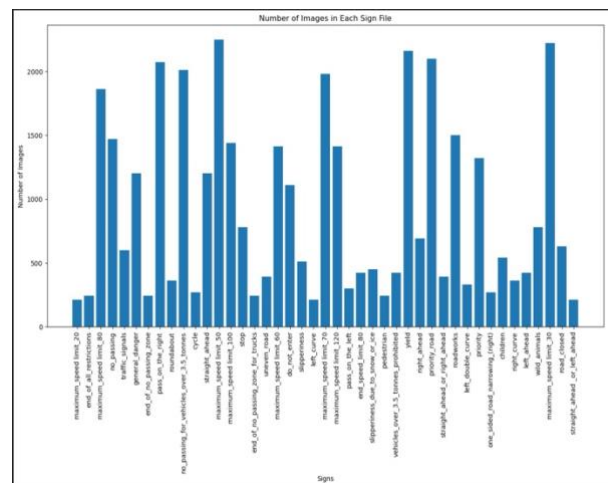
II. DATA PREPERATION

In the field of image recognition, where the quality and format of input data can greatly impact model performance, effective data preparation is essential to the success of any machine learning project. We used a large dataset from the INI

Benchmark Website for this traffic sign recognition research. This dataset was carefully selected to include a wide range of traffic sign photos taken in various environmental settings. This dataset offers a solid foundation for training our models since it contains differences in lighting, weather, angles, and occlusions.



Description of the Dataset: This collection of thousands of images is divided into 43 different classes, each of which represents a different type of traffic sign that one may see in both urban and rural areas. Every image is saved in the Portable Pixmap (PPM) format, which guarantees that no important visual data is lost during preprocessing and maintains the image quality. For the preprocessing requirements of this project, this format's great flexibility in manipulation and conversion makes it perfect.



Preprocessing Steps:

- Image Resizing: Every image was scaled to a standard 32 by 32 pixel size. The neural network's uniform input size is necessary to simplify its architecture and lower computational complexity without sacrificing the integrity of the picture data, and its standardization makes this possible.
- Normalization: Pixel values were normalized to a range of 0 to 1 in order to help the neural network train more quickly. In order to lessen the sensitivity of the model to the feature scale, the RGB values are normalized by going from a scale of 0-255 to a scale of 0-1.
- The final dataset was split as 80% and 20% for the training and validation sets, respectively. This division makes it possible for the model to be thoroughly trained on a significant quantity of data while also allowing for reliable validation on unbiased datasets to assess its performance.

III. MODEL ARCHITECTURE

In order to achieve efficient traffic sign identification, this project makes use of a combination of autoencoders and convolutional neural networks (CNNs). The architecture is intended to use an autoencoder to initially denoise and encode the traffic signs, followed by a CNN to classify the encoded features. The details of each part are listed below:

Autoencoder Architecture:

Input Layer: Receives the traffic sign images resized to 64x64 pixels in RGB format.

Encoder:

- Conv2D Layer: 16 filters with 3x3 kernels, ReLU activation, same padding.
- MaxPooling2D Layer: 2x2 pooling size, same padding.
- Conv2D Layer: 8 filters with 3x3 kernels, ReLU activation, same padding.
- MaxPooling2D Layer: 2x2 pooling size, same padding.
- Conv2D Layer: 8 filters with 3x3 kernels, ReLU activation, same padding.
- MaxPooling2D Layer: 2x2 pooling size, same padding (output encoded features).

Decoder:

- Conv2D Layer: 8 filters with 3x3 kernels, ReLU activation, same padding.
- UpSampling2D Layer: 2x2 upsampling size.
- Conv2D Layer: 8 filters with 3x3 kernels, ReLU activation, same padding.
- UpSampling2D Layer: 2x2 upsampling size.
- Conv2D Layer: 16 filters with 3x3 kernels, ReLU activation, same padding.
- UpSampling2D Layer: 2x2 upsampling size.
- Conv2D Layer: 3 filters with 3x3 kernels, Sigmoid activation, same padding (output reconstructed images).

The binary cross-entropy loss function and Adam optimizer, meant for the image reconstruction task, are compiled with the autoencoder.

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 64, 64, 3)	0
conv2d (Conv2D)	(None, 64, 64, 16)	448
max_pooling2d (MaxPooling2D)	(None, 32, 32, 16)	0
conv2d_1 (Conv2D)	(None, 32, 32, 8)	1,168
max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 8)	0
conv2d_2 (Conv2D)	(None, 16, 16, 8)	584
max_pooling2d_2 (MaxPooling2D)	(None, 8, 8, 8)	0
conv2d_3 (Conv2D)	(None, 8, 8, 8)	584
up_sampling2d (UpSampling2D)	(None, 16, 16, 8)	0
conv2d_4 (Conv2D)	(None, 16, 16, 8)	584
up_sampling2d_1 (UpSampling2D)	(None, 32, 32, 8)	0
conv2d_5 (Conv2D)	(None, 32, 32, 16)	1,168
up_sampling2d_2 (UpSampling2D)	(None, 64, 64, 16)	0
conv2d_6 (Conv2D)	(None, 64, 64, 3)	435

Total params: 4,963 (19.39 KB)

Trainable params: 4,963 (19.39 KB)

Non-trainable params: 0 (0.00 B)

Convolutional Neural Network (CNN) Architecture:

- Input Layer: Encoded features from the encoder part of the autoencoder.
- Flatten Layer: Flattens the output to form a one-dimensional vector.
- Dense Layer: 512 neurons, ReLU activation.
- Dense Layer: 256 neurons, ReLU activation.
- Dense Layer: 128 neurons, ReLU activation.
- Output Layer: 43 neurons (equal to the number of traffic sign classes), Softmax activation.

For multi-class classification problems, the CNN is constructed using the Adam optimizer and the categorical cross-entropy loss function. With this architecture, traffic sign identification under a variety of scenarios is robustly solved by combining the feature extraction capabilities of autoencoders with the classification capacity of CNNs.

IV. TRAINING PROCESS

The two primary phases of the Traffic Sign Recognition System's training process are the autoencoder and the convolutional neural network (CNN), which represent the two main parts of the model architecture. Every step has particular parameters and techniques designed to maximize efficiency.

Training the Autoencoder:

- Objective: The goal of the first stage is to train the autoencoder to learn effective ways to represent the images of traffic signs, with a particular emphasis on denoising and compressing the visual input.

- **Data Preprocessing:** The ImageDataGenerator class from Keras is used to preprocess the images. It does this by rescaling the pixel values to a range of 0 to 1. In addition, during training, this generator randomly modifies the photos, applying rotations, shifts, and flips, among other transformations, to enhance the data.
- **Network Architecture:** Using convolutional and pooling layers for encoding and convolutional and upsampling layers for decoding, the autoencoder is constructed in the manner outlined in the model architecture section.
- **Loss Function:** The loss function, binary cross-entropy, is appropriate for the binary pixel-wise classification that the autoencoder's reconstruction task requires.
- **Optimizer:** The Adam optimizer, its adaptive gradient update rules and efficient computing make it a popular choice for many deep learning applications, is used, with a learning rate usually set at 0.001.
- **Batch Size and Epochs:** A maximum of 100 epochs are allotted for the training process, which begins with a batch size of 32 for the model. To increase training efficiency and avoid overfitting, an early stopping mechanism is used to end training if the validation loss does not improve for three consecutive epochs.
- **Callbacks:** To keep an eye on the training process, EarlyStopping is employed as a callback. By terminating the training early if the model stops improving, this callback makes a major contribution to the optimization of the computational resources.

Training the CNN:

- **Objective:** The CNN is used in the second training phase to categorize the encoded features into the appropriate traffic sign categories.
- **Feeding Encoded Features:** All input images are converted into encoded features using the encoder portion (first half) of the autoencoder once it has been trained. The CNN is subsequently trained using these features as inputs.
- **Network architecture:** The CNN is built using the previously described design, which includes several dense layers that follow the encoder's flat input.
- **Loss Function:** For multi-class classification problems where each class is mutually exclusive, the CNN's loss function of categorical cross-entropy is selected.
- **Optimizer:** The Adam optimizer, which maintains the same learning rate for consistency and efficient learning rate adaption, is utilized in this instance, much as the autoencoder.
- **Batch Size and Epochs:** In order to prevent overfitting, the CNN is also trained using batches of 32 images spread over a possible 100 epochs. Early stopping is applied based on the performance on the validation set.
- **Validation Strategy:** Throughout CNN training, performance is continuously tracked for both training and validation. Apart from the training set, the

validation set offers an objective assessment of the model at every epoch.

Performance Evaluation:

- **Training and Validation Curves:** The loss and accuracy metrics are tracked during the autoencoder and CNN's training phases. Plotting these measures against the epoch number makes the learning curves visible and aids in determining if the model has begun to overfit or underfit.
- **Final Metrics:** To make sure the model generalizes successfully to new, unknown data, the model's performance is evaluated at the end of training using accuracy, precision, recall, and F1-score on a held-out test set.

V. RESULTS:

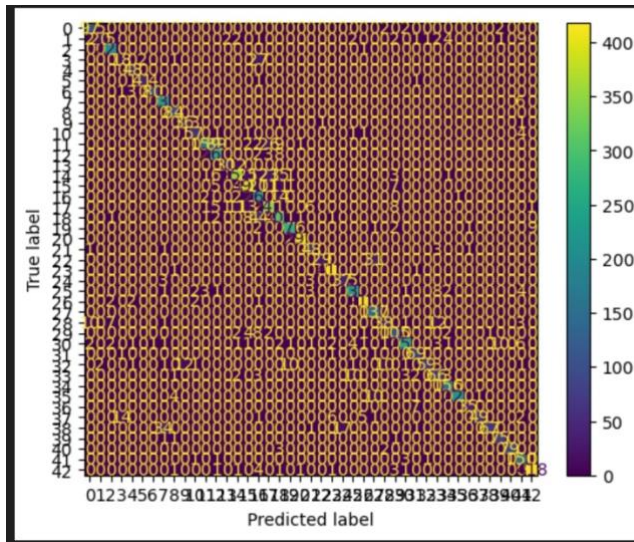
Several metrics were used to evaluate the accuracy and robustness of the Traffic Sign Recognition System in classifying traffic signs. The outcomes are essential in proving the system's dependability in a variety of scenarios.

Performance Metrics:

- **Accuracy:** On the test dataset, the model's overall accuracy was 88%. At this level of accuracy, the model is performing competently; most of the test set's traffic signs were accurately identified.
- **Precision, Recall, and F1-Score:** To give a thorough evaluation of the model's performance, precision, recall, and F1-scores were computed for each class. These measures are crucial for assessing how well the model predicts each class without favoring labels that occur more frequently. High precision and recall were found for often occurring indicators in the results, although certain less common signs needed improvement.

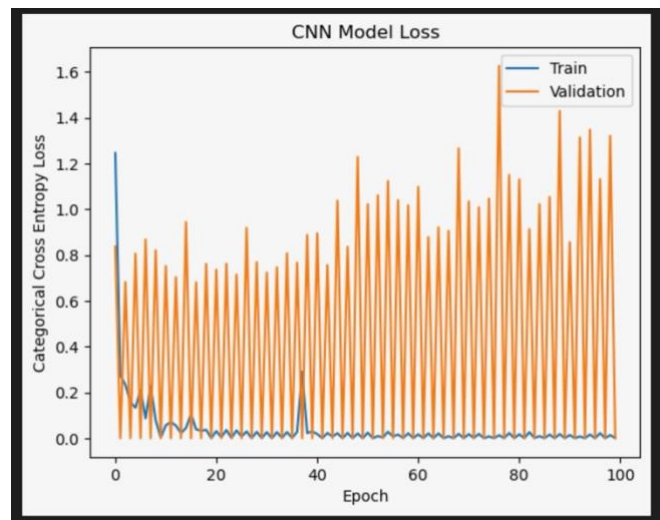
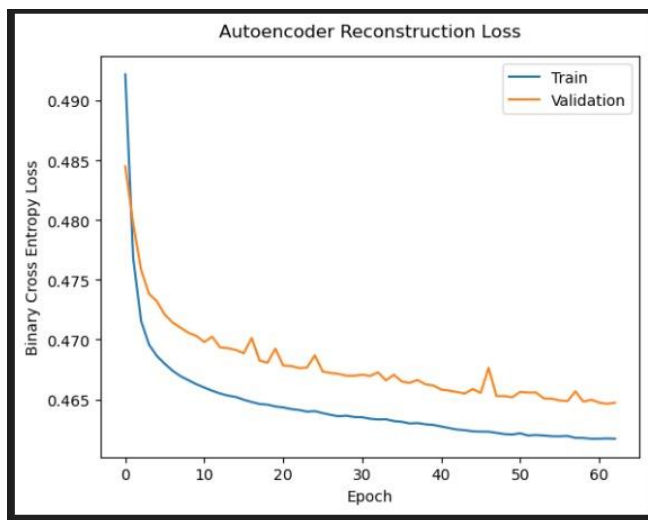
Classification Report:				
	precision	recall	f1-score	support
0	0.87	0.90	0.88	108
1	0.74	0.37	0.49	54
2	0.93	1.00	0.96	222
3	0.53	0.38	0.44	48
4	0.86	1.00	0.92	48
5	0.95	0.85	0.90	48
6	0.94	0.95	0.95	84
7	0.84	0.97	0.91	240
8	0.90	1.00	0.95	84
9	0.73	0.86	0.79	42
10	0.86	0.86	0.86	66
11	0.94	0.55	0.70	288
12	0.71	0.95	0.81	282
13	0.88	0.71	0.79	42
14	0.91	0.80	0.85	444
15	0.87	0.87	0.87	450
16	0.67	0.93	0.77	282
17	0.83	0.88	0.85	396
18	0.83	0.83	0.83	372
19	0.94	0.93	0.94	294
20	0.98	0.99	0.99	402
21	0.77	0.89	0.83	54
22	0.97	0.48	0.64	60
23	0.97	1.00	0.98	414
24	0.69	0.77	0.73	48
25	0.91	0.91	0.91	264
26	0.98	0.98	0.98	420
27	0.76	1.00	0.86	137
28	0.72	0.53	0.61	72
29	0.80	0.83	0.82	126
30	0.96	0.86	0.91	300
31	0.83	0.96	0.89	72
32	0.97	0.58	0.72	102
33	0.68	0.73	0.71	90
34	0.92	1.00	0.96	156
35	1.00	0.93	0.96	240
36	0.95	0.83	0.89	42
37	0.98	0.63	0.77	78
38	0.99	0.56	0.71	120
39	0.85	0.96	0.90	78
40	1.00	0.94	0.97	84
41	0.81	0.96	0.88	156
42	0.98	0.97	0.97	432
accuracy			0.88	7841
macro avg	0.86	0.83	0.84	7841
weighted avg	0.89	0.88	0.88	7841

- **Confusion Matrix:** To illustrate the model's performance over several classes and draw attention to the occurrences of both accurate and inaccurate classifications, a confusion matrix was employed. When it came to pinpointing specific classes—like closely related speed limit signs—where the model might mistake one sign for another, this matrix was especially helpful.



Visualization of Results:

Training and Validation Loss Curves: To keep an eye on the model's capacity for learning and generalization, graphs showing the training and validation losses throughout epochs were essential. Plots like these made it easier to determine when the model began to overfit or underfit.



Sample Predictions:

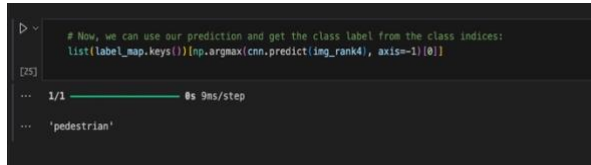
Using a particular traffic sign image from the dataset, a real-world test was carried out to confirm the model's efficacy in reality. The purpose of this test was to show that the model could correctly identify each of the various images that it had been trained to identify.

An image of a pedestrian crossing sign located in `"/Users/sunilinus/Desktop/DSP/Project/GTSRB/Final_Training/Images/pedestrian/00002_00008.ppm"` was selected as the test sample. This picture was preprocessed to comply with the model's input specifications. It was color-normalized and scaled to 64 by 64 pixels. To predict the traffic sign class, the processed image was fed into a CNN model that had been trained. The likelihood that the image belongs to each traffic sign class specified in the dataset is shown by the probability distribution the model produces across all potential classes.



In essence, the predictor accurately identified the image as a pedestrian crossing sign and produced a high confidence score for the 'Pedestrian' class. This attests to the model's successful acquisition of the visual characteristics linked to this traffic sign from the training set. To give a clear visual reference of what the model was evaluating and its prediction accuracy, the test involved displaying both the original image and its predicted output. The model's numerical prediction was converted back into a meaningful class label using the class indices from the training generator, validating the prediction that the class is "pedestrian." Making the

model's outputs comprehensible and applicable in real-world situations requires completing this phase.



```
# Now, we can use our prediction and get the class label from the class indices:
list(label_map.keys())[np.argmax(cnn.predict(img_rank4), axis=-1)[0]]

... 1/1 ----- 8s 9ms/step
... 'pedestrian'
```

This test provides an understanding of the operational effectiveness of the model and an actual illustration of how the Traffic Sign Recognition System can be used in real-world scenarios. Such individual testing are essential to comprehending and establishing confidence in the decision-making process of the model in real-world settings.

VI. CONCLUSION

The Traffic Sign Recognition System is an important development in the use of machine learning technology for autonomous vehicle systems and automobile safety. By employing an advanced combination of autoencoders and convolutional neural networks (CNNs), the system exhibits remarkable precision in identifying diverse traffic signs under varying environmental circumstances.

Achievements:

- **High Accuracy:** The system's overall accuracy of 88.0% on the test dataset demonstrates its reliability and robustness in correctly identifying traffic signs.
- **Effective Feature Extraction:** The clarity of the reconstructed images and the finely detailed features obtained demonstrate how successful the usage of autoencoders for denoising and feature extraction proved to be. These features were important for the ensuing classification tasks.
- **Advanced Classification with CNNs:** The CNN component demonstrated the capability of deep learning techniques in managing complex image recognition jobs by successfully classifying signs into 43 different categories.

Implications:

- **Enhanced Road Safety:** The system can potentially reduce traffic accidents caused by human error in sign recognition by effectively detecting traffic signs. This can be achieved by making a major contribution to the safety features of driver assistance systems and autonomous vehicles.
- **Scalability and Adaptability:** The architecture showed that it could handle an enormous data set that included a wide variety of traffic sign pictures. Because of its adaptability, it may be applied to further visual recognition tasks related to automobile technology.
- **Potential for Real-World Application:** If this type of technology is successfully implemented in real-world situations, it may contribute to the development of smart transportation systems, which may then be

integrated with other elements of smart city infrastructures in a seamless manner.

Future Work:

- **Dataset Expansion:** Adding more diverse data, particularly indicators under difficult conditions like continuous rain or snow, could strengthen the system's resilience.
- **Real-Time Processing:** The model's integration into live vehicle systems, where fast reaction times are essential, may be made easier by optimizing it for real-time processing on edge devices.
- **Integration with Other Systems:** A more complete system for vehicle navigation and decision-making may result from combining traffic sign recognition with other sensory inputs, such as radar and GPS.
- **Exploration of New Architectures:** Using sophisticated methods like Transfer Learning or looking into alternative neural network architectures like Capsule Networks may enhance accuracy and efficiency.
- **Addressing Class Imbalance:** The model's performance may be enhanced by putting strategies in place to manage class imbalance more effectively in the training set, particularly for less common signs.

In Conclusion, our effort has effectively shown how deep learning can be used to improve systems for recognizing traffic signs, which are essential parts of current autonomous driving systems. The approaches used—from data preparation to model evaluation—have not only produced encouraging outcomes but also established a model for further research into more advanced methods of machine learning for practical automotive applications.

VII. REFERENCES:

- [1] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel - "The German Traffic Sign Recognition Benchmark: A multi-class classification competition," in IEEE International Joint Conference on Neural Networks, 2011. This paper introduces the GTSRB dataset, a popular benchmark for traffic sign recognition systems.
- [2] TensorFlow Documentation - Official documentation and tutorials by TensorFlow are invaluable for practical guidance on implementing various neural network architectures.
- [3] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing and C. Igel, "Detection of traffic signs in real-world images: The German traffic sign detection benchmark," The 2013 International Joint Conference on Neural Networks (IJCNN)
- [4] Stallkamp, Johannes et al. "The German Traffic Sign Recognition Benchmark: A multi-class classification competition." The 2011 International Joint Conference on Neural Networks (2011): 1453-1460.
- [5] L. Jöckel, M. Kläs and S. Martínez-Fernández, "Safe Traffic Sign Recognition through Data Augmentation for Autonomous Vehicles Software," 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), Sofia, Bulgaria, 2019, pp. 540-541, doi: 10.1109/QRS-C.2019.00114.