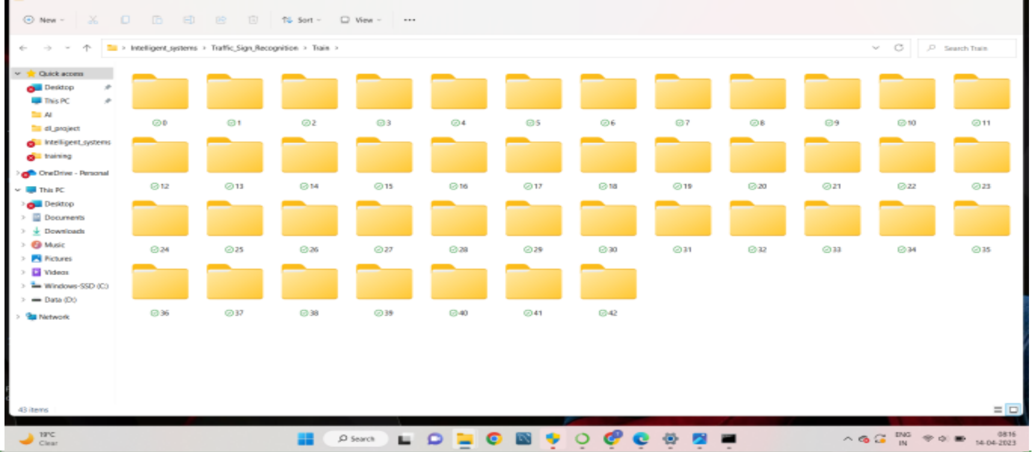
Contents:

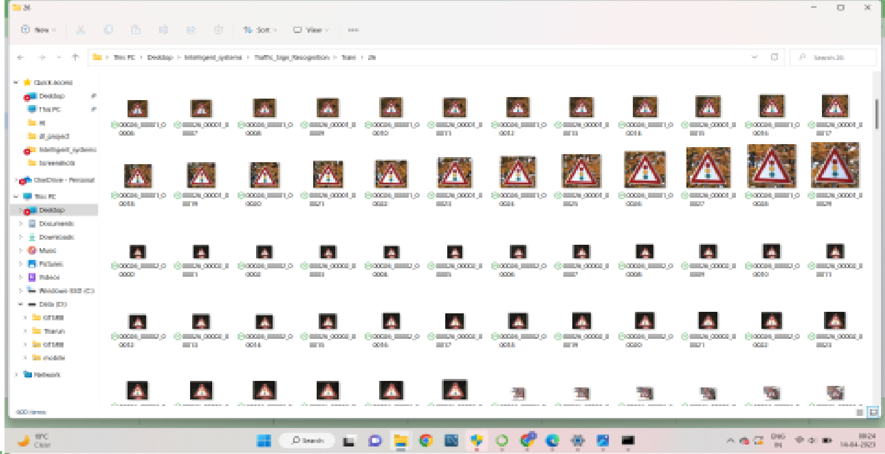
Introduction

- Traffic sign recognition is an important area of research in computer vision, with applications in self-driving cars, smart cities, and other areas where accurate and reliable identification of traffic signs is critical. In this project, we aim to build a deep learning model using convolutional neural networks to classify traffic signs from the German Traffic Sign Recognition Benchmark dataset.
- Our project involves several key steps, including preprocessing the dataset, designing and training a CNN model, evaluating the performance of the model on a validation set, testing the final model on a test set, and visualizing the model's activations to better understand how it is making predictions. By the end of the project, we hope to have a highly accurate and reliable traffic sign classification model that can be used in real-world applications.
- You Only Look Once (YOLO): We actually wanted to try YOLO which is an object detection algorithm but later we came to know that YOLO is realtime algorithm which can detect objects in real-time by processing the entire image in a single forward pass of a convolutional neural network. YOLO is fast and accurate and has been used in various computer vision tasks, including traffic sign detection.

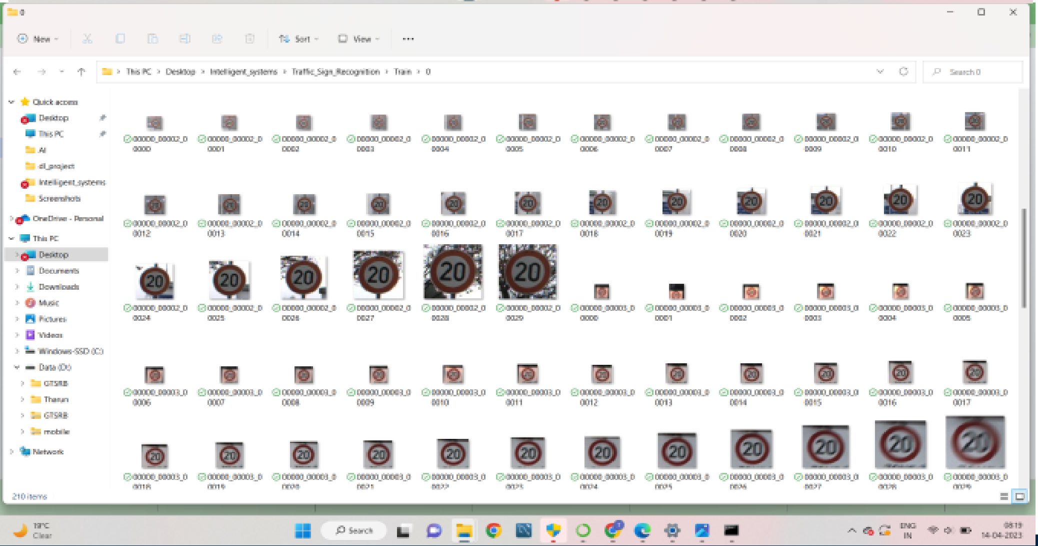
Data Description

- The German Traffic Sign Recognition Benchmark (GTSRB) dataset comprises of 39,209 training images and 12,630 test images for a total of 43 classes of traffic signs.
- The images are colored and have a resolution of 32x32 pixels. Red, green, and blue color channels are used to represent each pixel in the images.80% of the data is used for training, 10% is used for validation, and 10% is used for testing.
- The dataset is split into training, validation, and test sets. Each image has a class ID attached to it that stands for one of the 43 classes of traffic signs.
- Speed limit signs, stop signs, yield signs, no entrance signs, and pedestrian crossing signs are a few instances of the various traffic sign classes in the collection.

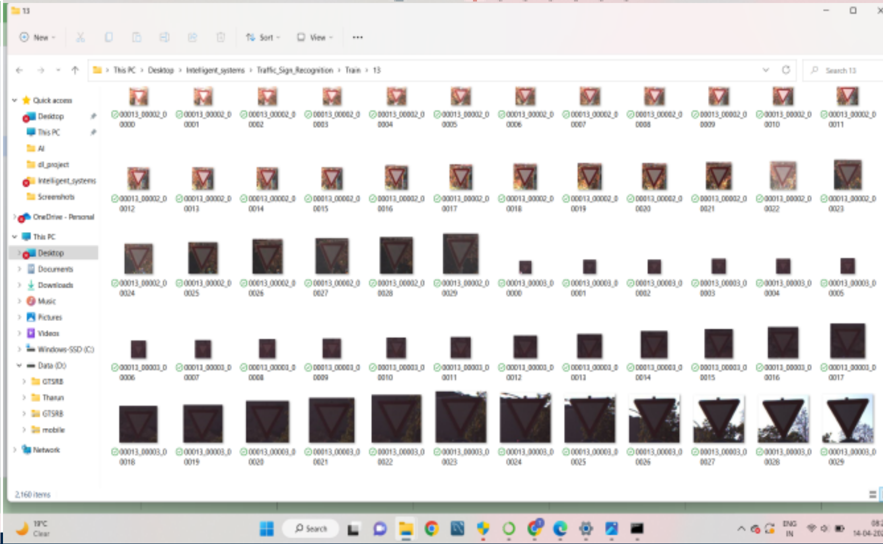




This PC > Desktop > Intelligent_systems > Traffic_Sign_Recognition > Train > 26



This PC > Desktop > Intelligent_systems > Traffic_Sign_Recognition > Train > 0
00000_00002_00000
00000_00003_00000
210 items



This PC > Desktop > Intelligent_systems > Traffic_Sign_Recognition > Train > 13
00013_00002_00000
00013_00003_00000
2,160 items

UNIVERSITY OF MICHIGAN

Background

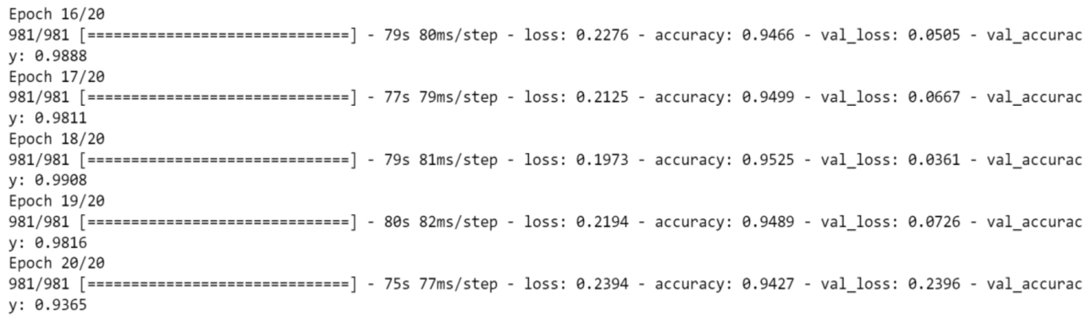
- In the field of computer vision, recognizing traffic signs is a crucial task with applications in driverless vehicles, traffic monitoring systems, and other intelligent transportation systems. The safety of drivers, pedestrians, and other road users depends on accurate and prompt recognition of traffic signs.
- Traffic sign identification techniques in the past relied on manually created features and machine learning algorithms, which frequently needed intensive feature engineering and manual tuning. Convolutional neural networks (CNNs) have, on the other hand, become a potent tool for picture classification tasks, such as traffic sign identification, thanks to recent developments in deep learning techniques.
- Without the requirement for explicit feature engineering, CNNs can automatically build hierarchical representations of visual characteristics from raw pixel data. Since the appearance and shape of traffic signs can differ greatly depending on elements like weather, illumination, and viewing angle, this makes them ideal for traffic sign recognition.
- The GTSRB dataset and other benchmarks for traffic sign identification have been the subject of numerous experiments using CNNs to attain cutting-edge results. We can contribute to the ongoing research in this field and create a highly accurate and dependable model that can be applied in real-world applications by creating a CNN model to categorize traffic signs from the GTSRB dataset.

Methodology

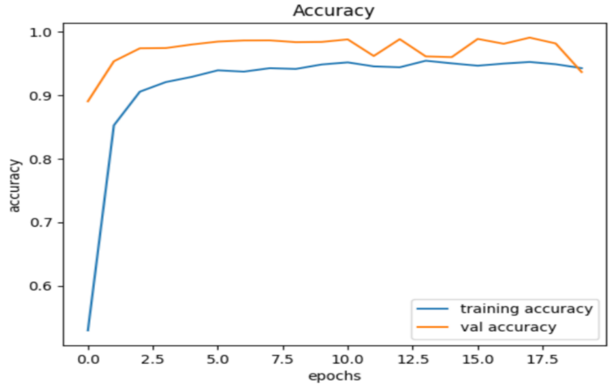
- Without the requirement for explicit feature engineering, CNNs can automatically build hierarchical representations of visual characteristics from raw pixel data. Since the appearance and shape of traffic signs can differ greatly depending on elements like weather, illumination, and viewing angle, this makes them ideal for traffic sign recognition.
- The GTSRB dataset and other benchmarks for traffic sign identification have been the subject of numerous experiments using CNNs to attain cutting-edge results. We can contribute to the ongoing research in this field and create a highly accurate and dependable model that can be applied in real-world applications by creating a CNN model to categorize traffic signs from the GTSRB dataset.
- The steps involved in our project are:

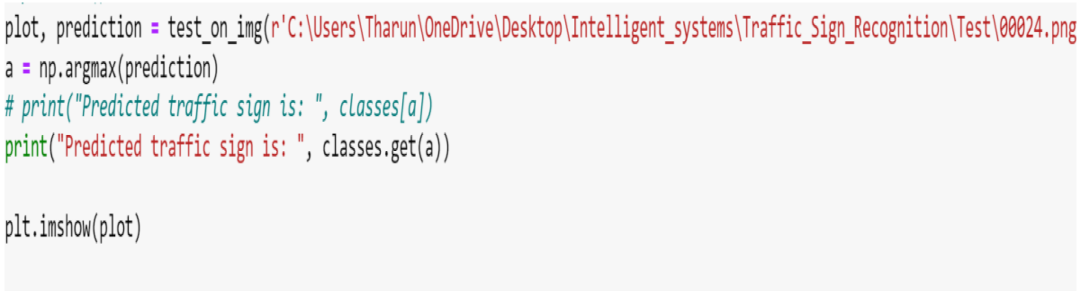
1. Data Collection
2. Data Preprocessing
3. Data Augmentation
4. Data Splitting
5. Model Architecture
6. Model Training
7. Model Evaluation
8. Model Fine-tuning

Results for first model

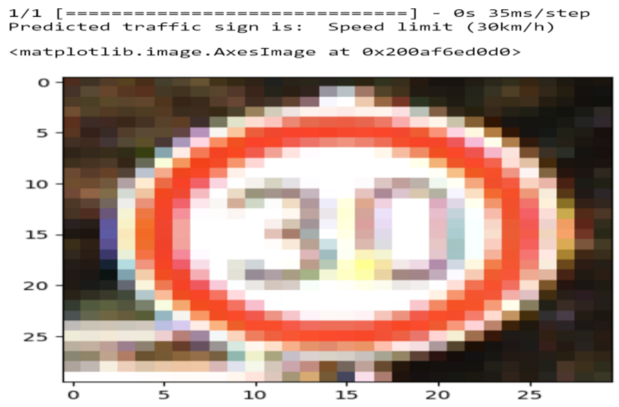


```
Epoch 16/20
981/981 [==============================] - 79s 80ms/step - loss: 0.2276 - accuracy: 0.9466 - val_loss: 0.0505 - val_accurac
y: 0.9888
Epoch 17/20
981/981 [==============================] - 77s 79ms/step - loss: 0.2125 - accuracy: 0.9499 - val_loss: 0.0667 - val_accurac
y: 0.9811
Epoch 18/20
981/981 [==============================] - 79s 81ms/step - loss: 0.1973 - accuracy: 0.9525 - val_loss: 0.0361 - val_accurac
y: 0.9908
Epoch 19/20
981/981 [==============================] - 80s 82ms/step - loss: 0.2194 - accuracy: 0.9489 - val_loss: 0.0726 - val_accurac
y: 0.9816
Epoch 20/20
981/981 [==============================] - 75s 77ms/step - loss: 0.2394 - accuracy: 0.9427 - val_loss: 0.2396 - val_accurac
y: 0.9365
```



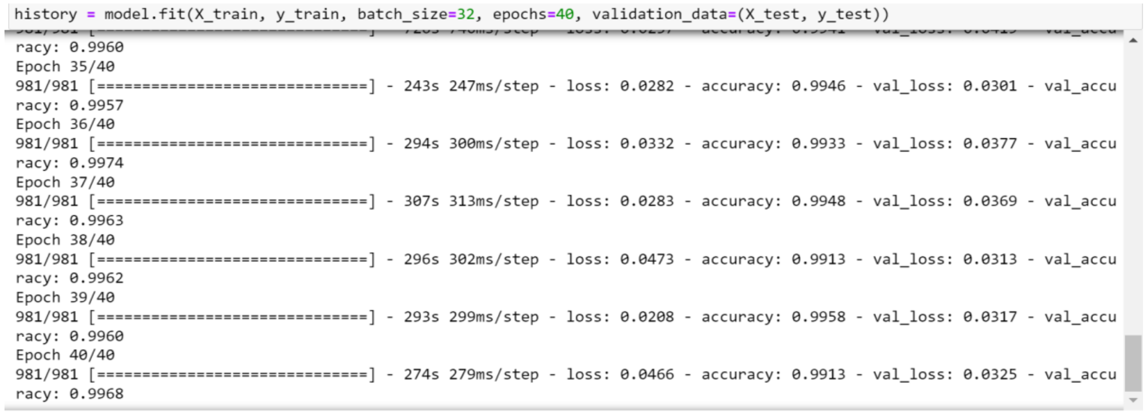


```
plot, prediction = test_on_img(r'C:\Users\Tharun\OneDrive\Desktop\Intelligent_systems\Traffic_Sign_Recognition\Test\00024.png
a = np.argmax(prediction)
# print("Predicted traffic sign is: ", classes[a])
print("Predicted traffic sign is: ", classes.get(a))

plt.imshow(plot)
```

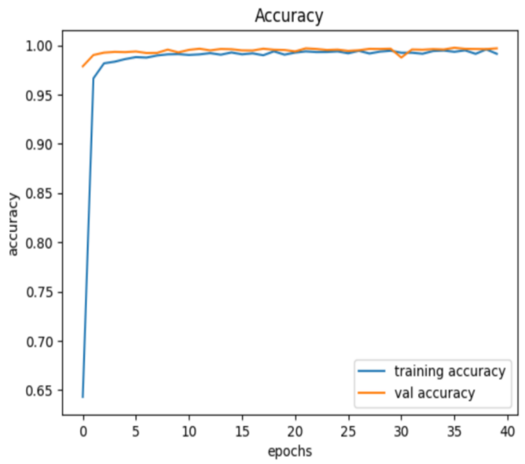


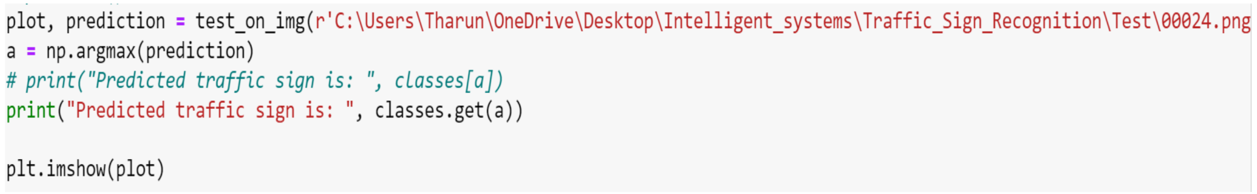
```
1/1 [==============================] - 0s 35ms/step
Predicted traffic sign is:  Speed limit (30km/h)

<matplotlib.image.AxesImage at 0x200af6ed0d0>
```

Results for second Model



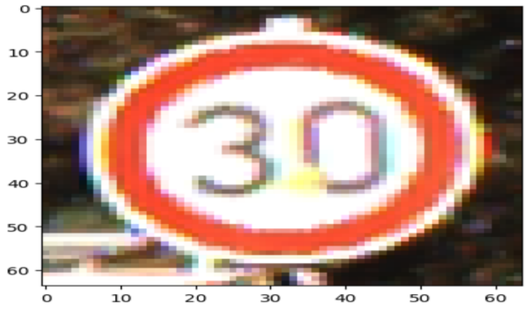
```
history = model.fit(X_train, y_train, batch_size=32, epochs=40, validation_data=(X_test, y_test))
racy: 0.9960
Epoch 35/40
981/981 [==============================] - 243s 247ms/step - loss: 0.0282 - accuracy: 0.9946 - val_loss: 0.0301 - val_accu
racy: 0.9957
Epoch 36/40
981/981 [==============================] - 294s 300ms/step - loss: 0.0332 - accuracy: 0.9933 - val_loss: 0.0377 - val_accu
racy: 0.9974
Epoch 37/40
981/981 [==============================] - 307s 313ms/step - loss: 0.0283 - accuracy: 0.9948 - val_loss: 0.0369 - val_accu
racy: 0.9963
Epoch 38/40
981/981 [==============================] - 296s 302ms/step - loss: 0.0473 - accuracy: 0.9913 - val_loss: 0.0313 - val_accu
racy: 0.9962
Epoch 39/40
981/981 [==============================] - 293s 299ms/step - loss: 0.0208 - accuracy: 0.9958 - val_loss: 0.0317 - val_accu
racy: 0.9960
Epoch 40/40
981/981 [==============================] - 274s 279ms/step - loss: 0.0466 - accuracy: 0.9913 - val_loss: 0.0325 - val_accu
racy: 0.9968
```





```
plot, prediction = test_on_img(r'C:\Users\Tharun\OneDrive\Desktop\Intelligent_systems\Traffic_Sign_Recognition\Test\00024.png
a = np.argmax(prediction)
# print("Predicted traffic sign is: ", classes[a])
print("Predicted traffic sign is: ", classes.get(a))

plt.imshow(plot)
```

```
1/1 [==============================] - 0s 167ms/step
Predicted traffic sign is:  Speed limit (30km/h)
<matplotlib.image.AxesImage at 0x133cf60ad00>
```



Discussion

- For our effort to recognize traffic signs, we employed a deep learning strategy called CNN (Convolutional Neural Network). The GTSRB (German Traffic Sign Recognition Benchmark) dataset, which has thousands of images of traffic signs tagged with their appropriate class labels, was used to train our CNN model.
- We created a CNN model with an input size of 30x30 pixels just for this assignment. Before training, we applied a number of pre-processing procedures to the photos, including scaling the image to a size of 60x60, augmentation, and normalization.
- Our model was able to catch more intricate details in the photographs by resizing the images, and augmentation broadened the diversity of our training data and minimized overfitting. The stability and convergence of our model during training were enhanced by normalization.
- In order to improve the performance of our network, we also changed the amount of layers and nodes. Our model's complexity increased when more layers and nodes were added, enabling it to capture photos with more intricate features, but it also increased the danger of overfitting and required more computational power to train.
- Overall, to get the greatest performance for our traffic sign recognition problem, our methodology combined pre-processing methods with careful model architecture and input size selection.

Conclusion

- In this research, we used convolutional neural networks and the GTSRB dataset to create a system for classifying traffic signs.
- In order to preprocess the photos, we first explored the dataset and resized and normalized the photographs.
- Then, utilizing a variety of hyperparameters and optimization strategies, we developed a CNN model with Keras and trained it on the dataset.
- In order to get better results, we assessed the model's performance using a variety of criteria, such as accuracy and precision, and then fine-tuned it using data augmentation approaches.

Future Work

- Future developments to the CNN-based traffic sign classification system are possible in a number of areas.
- Multi-modal data fusion is one potential area where additional data kinds, like LIDAR or radar, might be added to increase the system's accuracy and robustness.
- The quantity of labeled data required to train the model could be decreased by applying transfer learning as well. The system's architecture and hyperparameters might be optimized in real time to provide effective performance on systems with limited resources.
- New CNN architectures, including attention-based models or graph neural networks, could be tested to perhaps increase the system's precision and interpretability.
- Finally, strengthening the system's defenses against hostile attacks and learning how resilient it is to them could increase the system's dependability and security.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436-444, 2015.

[2] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.

[3] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv preprint arXiv:1804.02767, 2018.

[4] L. T. Tung and T. M. Nguyen, "Detecting Vietnamese Traffic Signs Using YOLOv3," in 2019 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), 2019, pp. 1-5. doi: 10.1109/CIVEMSA.2019.8801703.

[5] P. Sermanet and Y. LeCun, "Traffic sign recognition with multi-scale convolutional networks," in Proceedings of the International Joint Conference on Neural Networks (IJCNN), 2011, pp. 2809-2813

[6] S. Maldonado-Bascon, M. Marrón-Romera, and B. Guijarro-Berdiñas, "German Traffic Sign Detection and Recognition Using Convolutional Neural Networks," Sensors, vol. 19, no. 17, pp. 3732, 2019.

[7] D. Liang, X. Li, Y. Li, X. Yang, and Y. Li, "Traffic sign recognition based on convolutional neural network with improved data augmentation," Journal of Electronic Imaging, vol. 28, no. 3, pp. 033005, 2019.

UNIVERSITY OF MICHIGAN

THANK YOU