DLP PROJECT REPORT

TRAFFIC SIGN CLASSIFIER

by

20K-0241 Adil Pervez| BCS-8A

20K-0368 Hammad Shakeel | BCS-8A

20K-0225 Muhammad Umer | BCS-8A

Course Instructor: Miss Sumaiyah Zahid

Abstract:

This project entails the classification of images using both convolutional neural network (CNN) and ResNet models trained on the German Traffic Sign Benchmark dataset. With over 50,000 images categorized into more than 40 classes of traffic signs, this dataset provides a robust foundation for training deep learning models. The CNN and ResNet architectures are employed separately to classify traffic signs accurately, and their results are compared for performance evaluation.

Introduction and Motivation:

Recognizing traffic signs is pivotal for ensuring road safety and efficient traffic management. However, traditional methods for traffic sign recognition often struggle with real-world variations and complexities. To address this challenge, deep learning models such as CNNs and ResNet offer promising solutions by automatically learning and extracting meaningful features from images.

The motivation behind this project stems from the critical importance of accurate traffic sign classification in enhancing road safety. By leveraging the extensive German Traffic Sign Benchmark dataset, which encompasses a diverse array of traffic signs, the goal is to train robust classification models capable of handling real-world scenarios effectively.

Through the utilization of both CNN and ResNet architectures, this project aims to explore and compare their performance in traffic sign classification tasks. By providing insights into the strengths and weaknesses of each model, the project endeavors to contribute to the ongoing efforts aimed at improving road safety through advanced computer vision techniques. Ultimately, the overarching objective is to mitigate the risks associated with accidents caused by human error through more accurate and reliable traffic sign recognition 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.

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.

In ResNet, the initial preprocessing steps involve scaling the images to ensure pixel values fall within the range [0, 1], aiding in numerical stability during model training. Then, the images are prepared for input into the ResNet50 architecture, which inherently handles image resizing and processing through its convolutional layers.

Model:

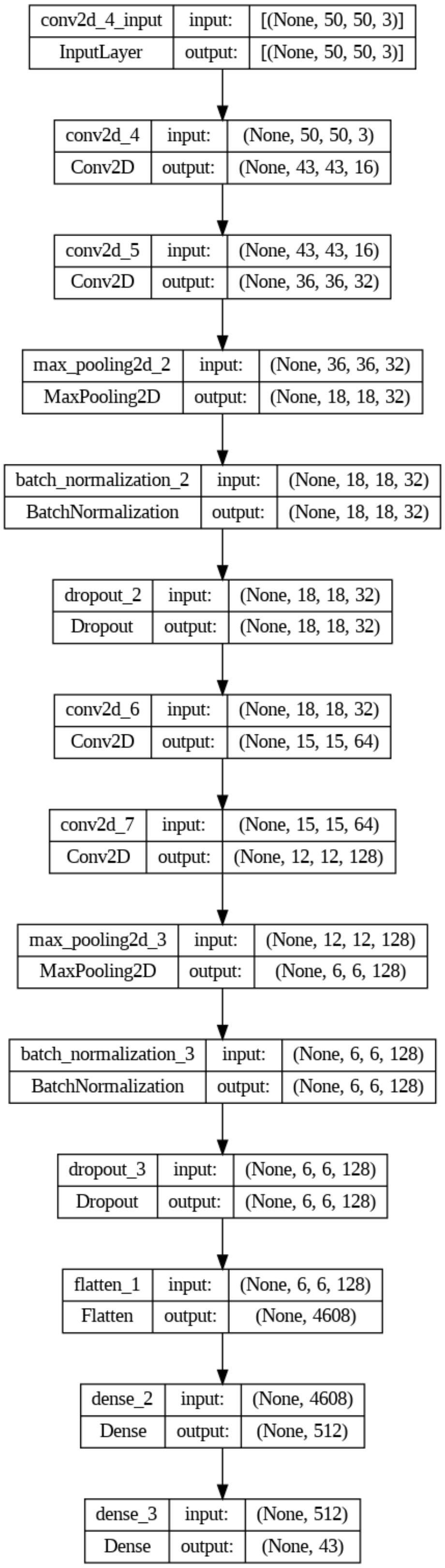
We picked a pre-trained sequential CNN model which had 95% accuracy however we decided to fine-tune the model by using different kernel sizes and no of filters at each convolution and added some additional layers and batch normalization as well.After every two convolutional layers Max pooling was applied 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.

For the ResNet model, a pre-trained ResNet50 architecture was employed, originally trained on the ImageNet dataset. Fine-tuning included adjustments to kernel sizes and the number of filters at each convolutional layer. Additionally, batch normalization layers were incorporated. Max pooling with a stride of 2 was applied after every two convolutional layers. Dropout layers were introduced to mitigate overfitting. ReLU activation function was used throughout the network to address vanishing gradients. These modifications and training iterations led to a testing accuracy of 99.26% and a training accuracy of 98.82%.

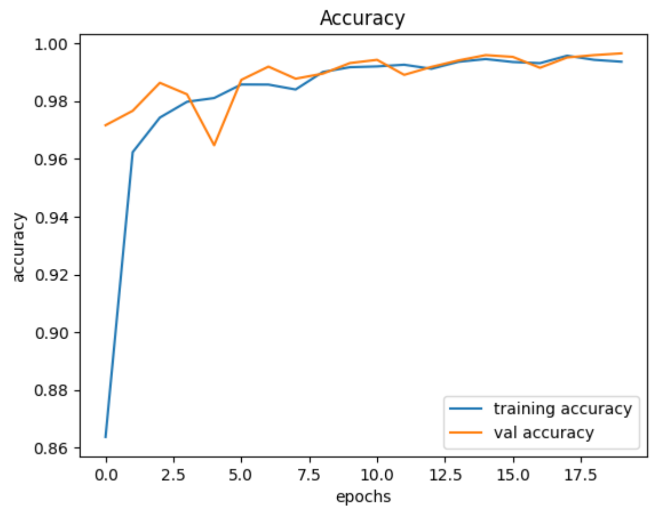
Training:

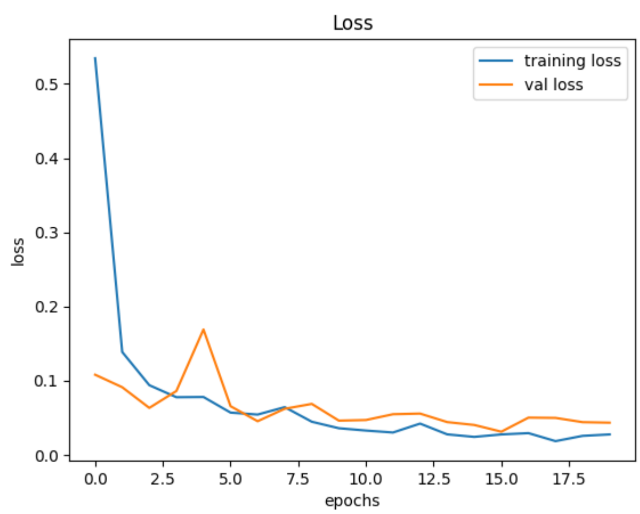
We trained the new model over the Train folder in the dataset that had 38000+ images with 2.5Million parameters by using the Gpu runtime on Google colab.We had a 75:25 testing to validation split and we ran 20 epochs with the batch size set to 32 which meant that each epoch contained over 981 samples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

The ResNet model's training process and performance evaluation were similar to the CNN model, albeit with variations in architecture and parameter configurations. After preprocessing and training, the ResNet model exhibited notable accuracy and robustness in classifying traffic signs, contributing to the project's overarching goal of enhancing road safety through advanced computer vision techniques. The model led to a testing accuracy of 99.26% and a training accuracy of 98.82%.

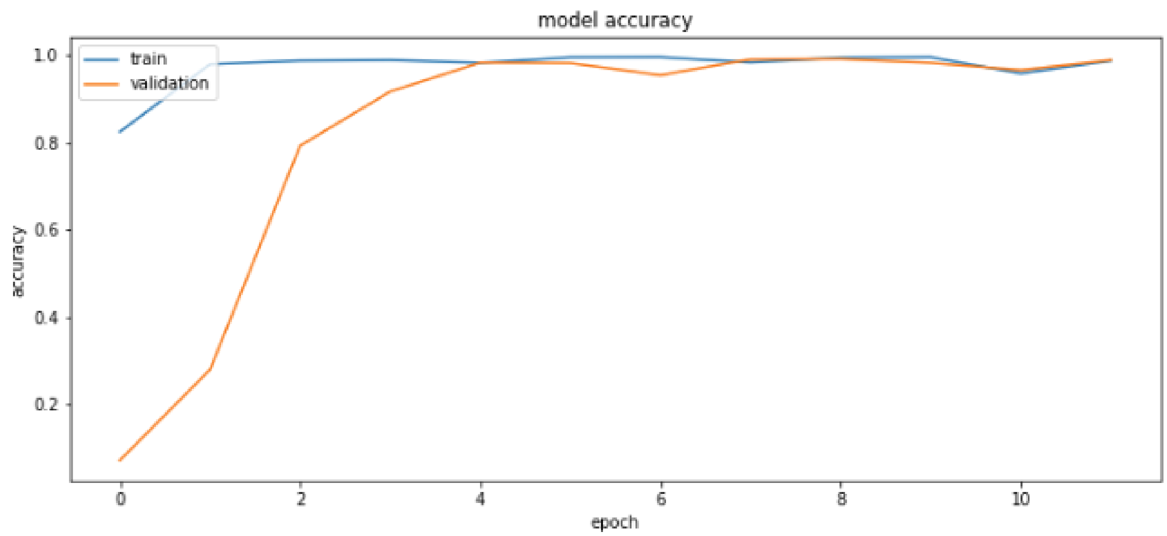

conv2d_4_input | InputLayer | input: [(None, 50, 50, 3)] | output: [(None, 50, 50, 3)]
conv2d_4 | Conv2D | input: (None, 50, 50, 3) | output: (None, 43, 43, 16)
conv2d_5 | Conv2D | input: (None, 43, 43, 16) | output: (None, 36, 36, 32)
max_pooling2d_2 | MaxPooling2D | input: (None, 36, 36, 32) | output: (None, 18, 18, 32)
batch_normalization_2 | BatchNormalization | input: (None, 18, 18, 32) | output: (None, 18, 18, 32)
dropout_2 | Dropout | input: (None, 18, 18, 32) | output: (None, 18, 18, 32)
conv2d_6 | Conv2D | input: (None, 18, 18, 32) | output: (None, 15, 15, 64)
conv2d_7 | Conv2D | input: (None, 15, 15, 64) | output: (None, 12, 12, 128)
max_pooling2d_3 | MaxPooling2D | input: (None, 12, 12, 128) | output: (None, 6, 6, 128)
batch_normalization_3 | BatchNormalization | input: (None, 6, 6, 128) | output: (None, 6, 6, 128)
dropout_3 | Dropout | input: (None, 6, 6, 128) | output: (None, 6, 6, 128)
flatten_1 | Flatten | input: (None, 6, 6, 128) | output: (None, 4608)
dense_2 | Dense | input: (None, 4608) | output: (None, 512)
dense_3 | Dense | input: (None, 512) | output: (None, 43)

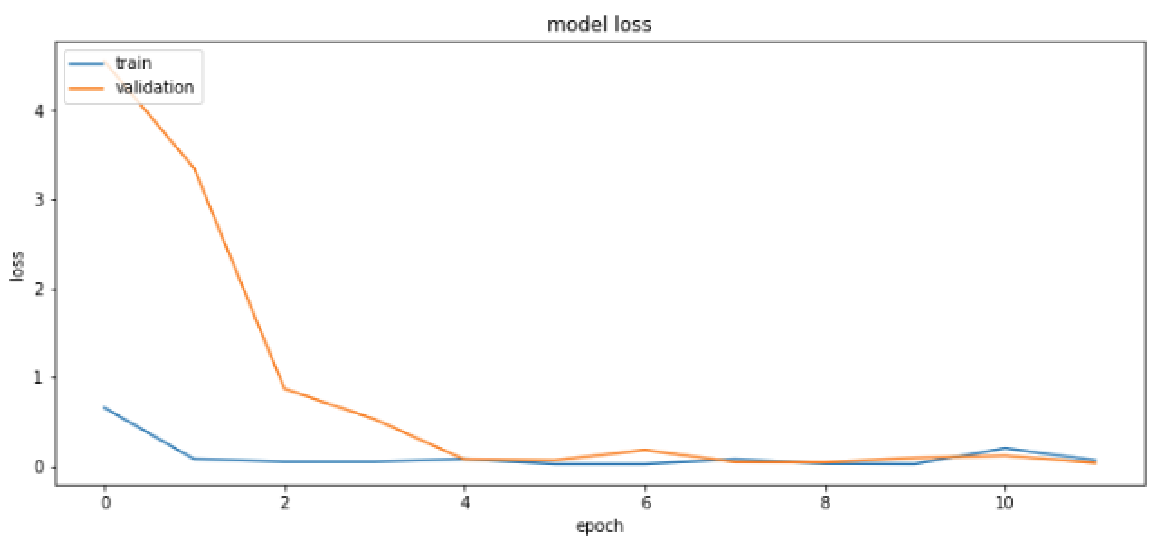
CNN:


Accuracy
accuracy
epochs
training accuracy
val accuracy


Loss
loss
epochs
training loss
val loss

Resnet:


model accuracy
train
validation
accuracy
epoch
1.0
0.8
0.6
0.4
0.2
0
2
4
6
8
10


model loss
train
validation
loss
epoch
4
3
2
1
0
0
2
4
6
8
10

Testing:

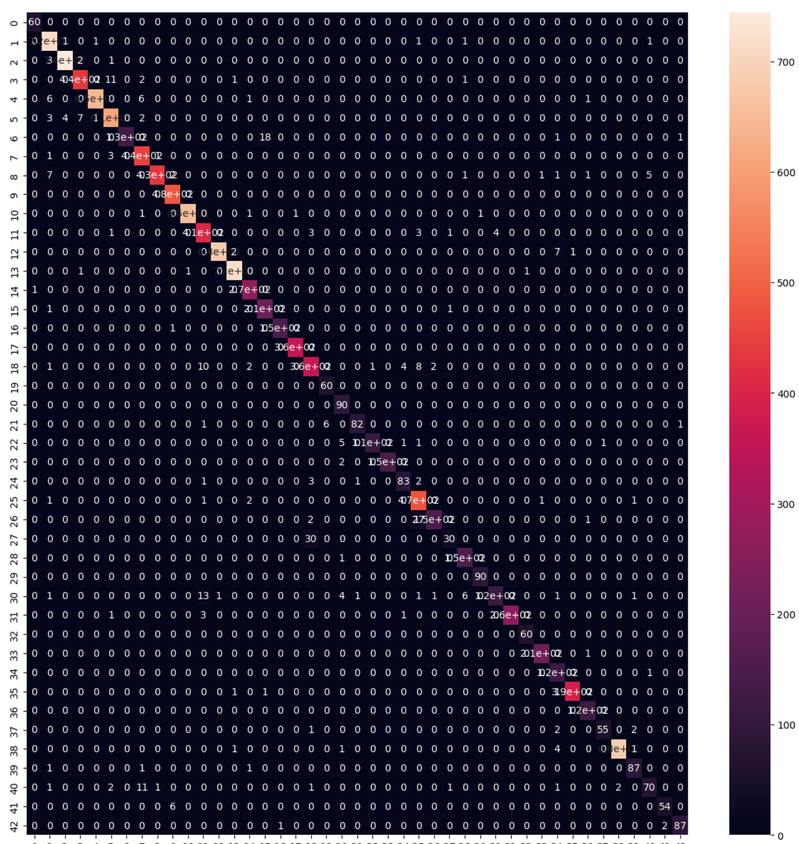
CNN:

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.

```python
#Accuracy with the test data
from sklearn.metrics import accuracy_score
print('Accuracy: ' + str(accuracy_score(labels, pred)*100) + ' %')
```

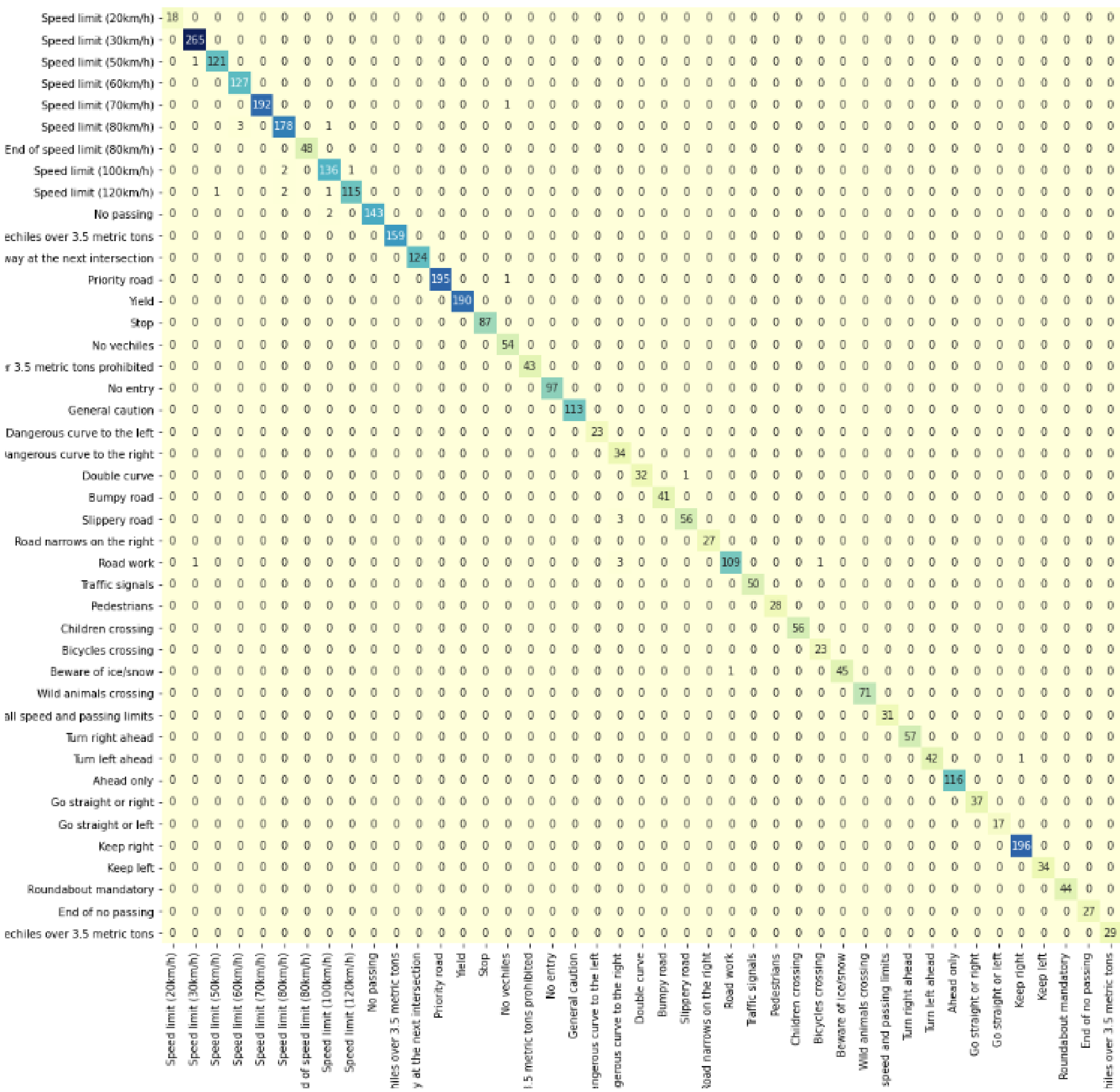
```
395/395 [==============================] - 1s 3ms/step
Accuracy: 97.33174980205858 %
```

Figure 4.



Resnet:

The confusion matrix visualization provides insights into the model's classification performance across different traffic sign categories. Each cell in the matrix represents the number of predictions for a given actual class label. The annotated heatmap aids in identifying any misclassifications or patterns in the model's predictions, contributing to a comprehensive assessment of its performance.



Results:

CNN:

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.

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

Resnet:


Road work
Traffic signals
Road work
Speed limit (50km/h)
Yield
Turn right ahead
Speed limit (120km/h)
Speed limit (60km/h)
Priority road
No passing
No passing
Right-of-way at the next intersection
Keep right
No passing
Dangerous curve to the right
Road work
Speed limit (70km/h)
Turn left ahead
Beware of ice/snow
Yield

Conclusion and Future Work:

The models, both CNN and ResNet, demonstrated promising results, showcasing their efficacy in accurately classifying traffic signs with high precision. Through training on a comprehensive dataset and leveraging sophisticated deep learning techniques, substantial enhancements in classification accuracy were achieved. The traffic sign classifier, powered by these models, holds practical implications for various domains such as autonomous vehicles, driver assistance systems, and traffic management.

The CNN-based traffic sign classifier exhibited notable accuracy, laying the groundwork for its potential deployment in real-world scenarios. Similarly, the ResNet model showcased remarkable performance, further bolstering the project's objectives of enhancing road safety through precise traffic sign recognition.

Future endeavors could focus on fine-tuning the models and exploring additional enhancements to address any residual challenges in real-world applications. One potential for improvement involves extending the system to support real-time data input, enabling the models to process live images captured by a camera. This real-time functionality would empower the classifier to swiftly analyze and classify traffic signs, offering instantaneous feedback and insights to support decision-making on the road.

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/.

Kakarla, S. (n.d.). Traffic Sign Classification Using Residual Networks (ResNet). Towards Data Science.https://towardsdatascience.com/traffic-sign-classification-using-residual-networks-resnet-4b550046ff83