Traffic Sign Classifier

Group Name - Data Diggers

PROJECT REPORT

Submitted by:

Divyansh Khandelwal (ID: 11840440)

Tanish Gupta (ID: 11841150)

Guided by:

Dr. Subidh Ali

INDIAN INSTITUTE OF TECHNOLOGY BHILAI

CS550 - Machine Learning

Abstract

Automatic traffic sign recognition is a field of deep learning which is a very important aspect for advanced driver support systems. This project proposes a framework that will detect and classify different types of traffic signs from GTSRB images dataset. CNN and image augmentation techniques are used in the classification model resulting in an accuracy greater than 95 percent. Finally a UI is created to test the images which have been created using the streamlit library which we will be trying to deploy in Amazon AWS cloud platform later.

INTRODUCTION

The development of advanced driver support systems leads to the birth of traffic sign detection and classification problems. In order to solve the concerns over road and transportation safety, automatic traffic sign recognition (TSR) systems have been introduced. An automatic TSR system can detect and recognise traffic signs from and within images captured by cameras or imaging sensors. In adverse traffic conditions, the driver may not notice traffic signs, which may cause accidents. In such scenarios, the TSR system comes into action.

For this project, we have created a deep learning classification model using CNN and image augmentation techniques on German Traffic Sign Recognition Benchmark (GTSRB) dataset which is providing an accuracy of greater than 95 percent. The dataset contains 43 classes of traffic signs. We have created a **web app to test images** and also **implemented a text to audio algorithm** which will output the result of classification in an audio format in the web app.

PROBLEM DEFINITION

Our Problem is of Traffic sign Classification where we need to predict the class of traffic sign that is given as input to us. Since the problem is too easy so in order to make it more complex we have created a web app using streamlit library ,so basically what we are doing is taking the input image using this streamlit library and then loading the saved model in our web app and then predicting the class our image belongs to.

And also we have **implemented a text to audio feature** which will output the result of classification in an audio format in the web app.

OBJECTIVE

This project can be used in automobiles like cars and trucks to recognize traffic signs along the roads so that the road users can be notified about regulations and provide warning and guidance needed for safe, uniform and efficient operation.

The traffic signs should be considered as a guide or a speaker on a road network because generally drivers ignore many of the traffic signs due to reasons like high speed of the vehicle or while overtaking some other vehicle on the road. This ignorance of traffic signs leads to accidents which can be avoided by implementing our project model since our project contains the audio features also.

DATASETS USED

During the past few years, a number of research groups have worked on creating traffic sign datasets for the task of detection, recognition, and tracking. One of these datasets are publicly available for use and we have used that dataset in our project and the dataset is **German Traffic Sign Recognition Benchmark (GTSRB).**

Country of origin: Germany
Number of images: 50000
Classes: 43.

Problems that we faced in our initial model and the model architecture and also the explanation of each model layer has been already done in our previous report. Now we are maintaining the updated part of our model that we have done and how that has led to improvement in our model accuracy.

UPDATED MODEL ARCHITECTURE :

Our model is adapted from LeNet by Yann LeCun. It is a convolutional neural network designed to recognize visual patterns directly from pixel images with minimal preprocessing. It can handle hand-written characters very well.

Source:

http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

The architecture of our updated model is given below

- 2 Conv2D layer (filter=32, **kernel_size=(3,3)**, activation="relu")
- MaxPool2D layer (pool_size=(2,2))
- Dropout layer (rate=0.25)
- 2 Conv2D layer (filter=64, kernel_size=(3,3), activation="relu")
- MaxPool2D layer (pool_size=(2,2))
- Dropout layer (rate=0.25)
- Flatten layer to squeeze the layers into 1 dimension
- Dense Fully connected layer (**512 nodes**, activation="relu")
- Dropout layer (rate=0.5)
- Dense layer (43 nodes, activation="softmax")

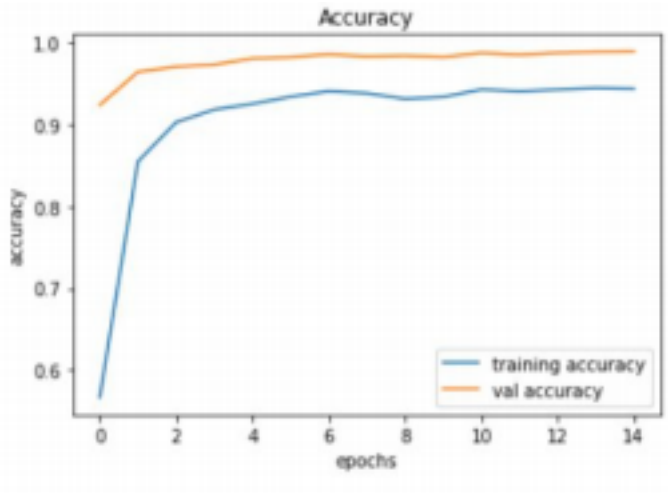
CHANGES WE INTRODUCED IN OUR NEW MODEL :

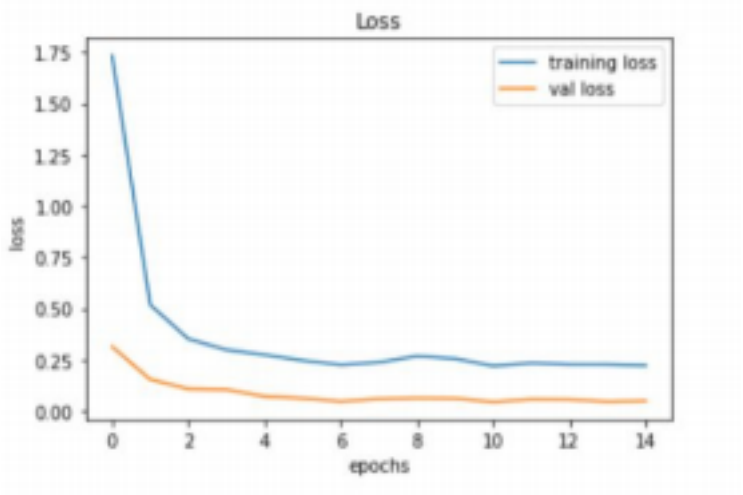
We worked on various model architectures to improve our model accuracy and performance, like by changing filters or varying dense layers or dropouts ,and then observed our model performance .After working on various model architectures we found that our model was giving good accuracy when we are **increasing the dense layer and decreasing the filter sizes**.

And the reason could be that, when we were using the larger kernel size we were losing some details in the smaller features while when we used a 3x3 filter it detected those features and thus resulted in better accuracy .
And also a model with more layers and more hidden units per layer has higher representational capacity i.e it is capable of representing more complicated functions. Thus , increasing the model capacity leads to improving our model performance.

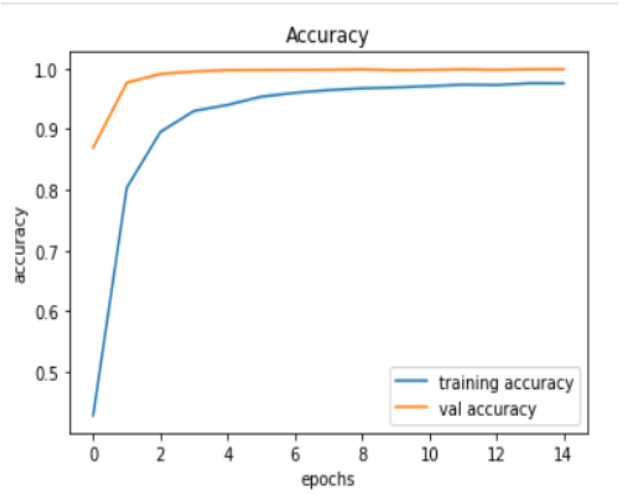
MODEL PERFORMANCE BEFORE AND AFTER THE CHANGES APPLIED:

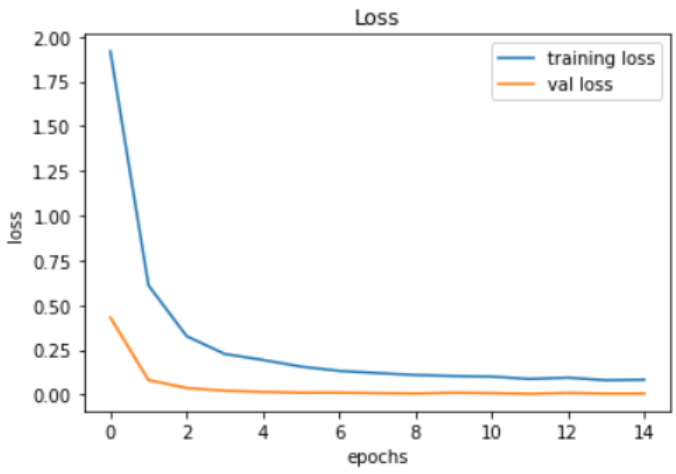
Before applying the changes in our model architecture, following is the plot of our model performance:





After applying the changes in our model architecture, following is the plot of our model performance:





we can clearly see that our model training accuracy/validation accuracy is improved and training loss/validation loss is decreased,thus our model has improved from the previous one.

MODEL LAYER SUMMARY:

Following is our updated model summary and since we had already written the details of the layer explanation in our last report ,we are not writing it again . Here we can clearly see the difference in the output shape of our model due to the changes in the filter sizes and dense layer nodes.

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 28, 28, 32)        896
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 26, 26, 32)        9248
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
dropout_9 (Dropout)          (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 11, 11, 64)        18496
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 9, 9, 64)          36928
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
dropout_10 (Dropout)         (None, 4, 4, 64)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_6 (Dense)              (None, 512)               524800
_________________________________________________________________
dropout_11 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_7 (Dense)              (None, 43)                22059
=================================================================
Total params: 612,427
Trainable params: 612,427
Non-trainable params: 0
```

RESULT OBTAINED IN PREVIOUS REPORT:

The testing accuracy on our test Dataset in our previous model was 96.088.

```
#testing accuracy on test dataset

from sklearn.metrics import accuracy_score

print('Test Data accuracy: ',accuracy_score(labels_test, pred)*100)
```

```
Test Data accuracy:  96.08867775138559
```

After Epoch=15 , Following is the **model performance** which we have observed :

Training loss: 0.1020, Training accuracy: 0.9706 -

Validation loss: 0.0097 , Validation Accuracy: 0.9976

UPDATED RESULT :

The testing accuracy on our test Dataset after model improvement is 97.7355.

```
#testing accuracy on test dataset

from sklearn.metrics import accuracy_score

print('Test Data accuracy: ',accuracy_score(labels_test, pred)*100)
```

```
Test Data accuracy:  97.73555027711798
```

After Epoch=15 , Following is the updated **model performance** which we have observed :

Training loss: 0.0892, Training accuracy: 0.9738 -

Validation loss: 0.0059 , Validation Accuracy: 0.9985

FINAL OUTPUT :

We have deployed our model using the streamlit library, In this interface we just need to upload an image and it will predict the class of that image and also generates an audio which will tell us what is the class of the image and from which class it belongs to.

REFERENCES:

For code reference for our **model** we have taken help from these links-

https://www.pyimagesearch.com/2019/11/04/traffic-sign-classification-with-keras-ad-deep-learning/

https://data-flair.training/blogs/python-project-traffic-signs-recognition/

Front-end code reference (streamlit part)

https://github.com/dD2405/Traffic_Sign_Classifier-with-Streamlit/blob/master/app.py

For adding **audio part** we have taken help from the **streamlit documentation** and then added the audio:

https://docs.streamlit.io/en/stable/api.html