“ Smart traffic control system with application of image processing techniques ”

→ **Traffic Sign Classification Using CNN**

Prepared by :

Passant Adel	20190134
Yasmine Saber	20190620
Hussien Hesham	20190183
Nour Eldin ezzat	20190593
Youssef Mohamed	20190642

ABSTRACT

As the demand for autonomous electric vehicles grows, there are numerous technical challenges. prospects within the structure that have a lot of scope for advancement. One such prospect is traffic sign recognition. In this project we implement feature classification using convolutional neural networks to achieve high efficiency and accuracy.

INTRODUCTION

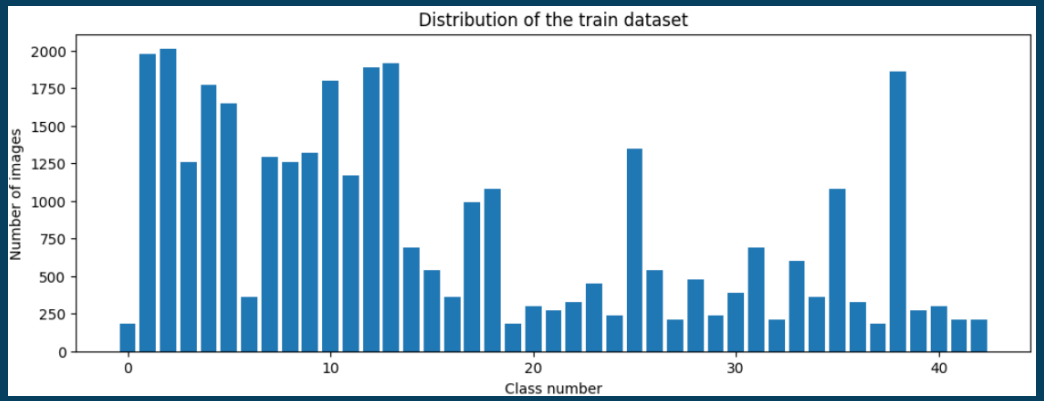
It is common knowledge that traffic sign classification is one of the most significant parts of an autonomous vehicle system because it helps to prevent accidents and regulate traffic flow on roads. As soon as driver-less vehicles came into play a couple of decades ago, traffic sign classification was one of the burning topics as a lot of research was spent into developing an ideal model. Initially, traditional computer vision and machine-based methods were used to implement the recognition model but seeing as they did not have enough accuracy and efficiency, they were soon replaced by deep learning-based classifiers.

Dataset

- DATASET German Traffic Sign Recognition Dataset (GTSRB) is an image classification dataset. The images are photos of traffic signs. The images are classified into 43 classes. The training set contains 39209 labelled images and the test set contains 12630 images.
- Dataset consist of traffic sign images.
- These images are classified in 43 classes.
- These categories are labelled from 0 - 42.
- Training Dataset consists of 39209 images.
- Test Dataset consists of 12630 image

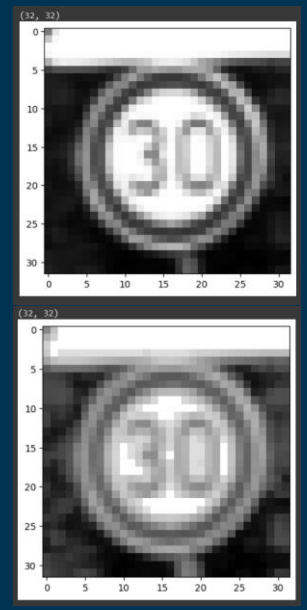
```
print(X_train.shape)
print(X_val.shape)
print(X_test.shape)
```

```
(34799, 32, 32, 3)
(4410, 32, 32, 3)
(12630, 32, 32, 3)
```



Preprocessing

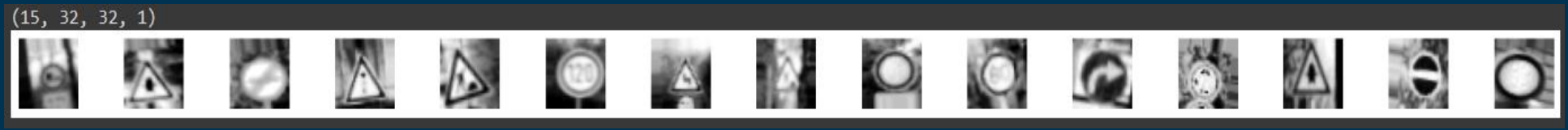
1. Convert the image to grayscale i.e. binary images with only two colours black or white. **Why Convert to grayscale?** , Well for traffic signs is colour really important? I don't think so, the important details are the **edges** and the **shapes** of the signs. Also We'll now need **fewer input parameters** for our neural network. Plus **less Computing Power**.



2. Next We apply the Histogram Equalization technique to standardize lighting in all our images. Well because some images are more brighter and others very dim. Thus We need to give them a similar lighting effect.

3. Next We add a depth to our images, that is required for our Convolutional Layer

4. We generate variety datasets to better the model. Not *generate* but we show different angles and views of the same data set to the model for it to better identify the features.

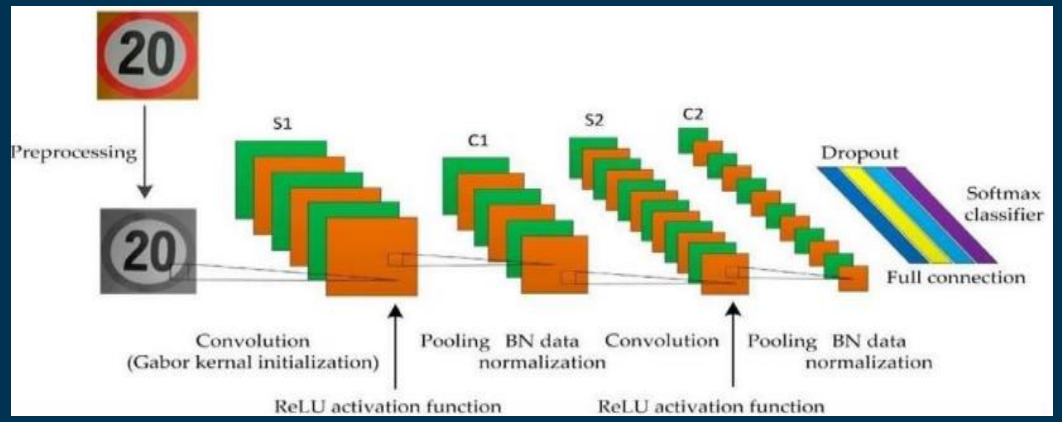


5. Next One Hot Encode the data labels. One hot encoding is a technique used to convert categorical data, such as class labels, into a numerical format that can be used as input to a machine learning model.

Model Architecture

Our Model Based On LeNet Architecture

The model consists of several layers, including convolutional layers, pooling layers, a dropout layer, and dense layers. The Conv2D layers perform convolution operations on the input images, with different filter sizes and activation functions. The MaxPooling2D layers perform pooling operations to reduce the spatial dimensions of the feature maps. The Flatten layer flattens the output of the convolutional layers into a 1-dimensional array. The Dense layers are fully connected layers that perform classification based on the extracted features. The Dropout layer is used to prevent overfitting by randomly dropping out some of the nodes during training. The model is compiled using the Adam optimizer with a learning rate of 0.001 and a categorical cross-entropy loss function.

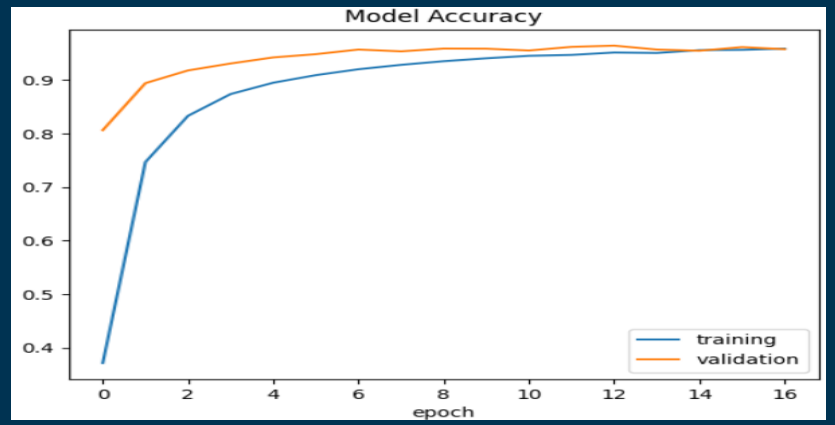


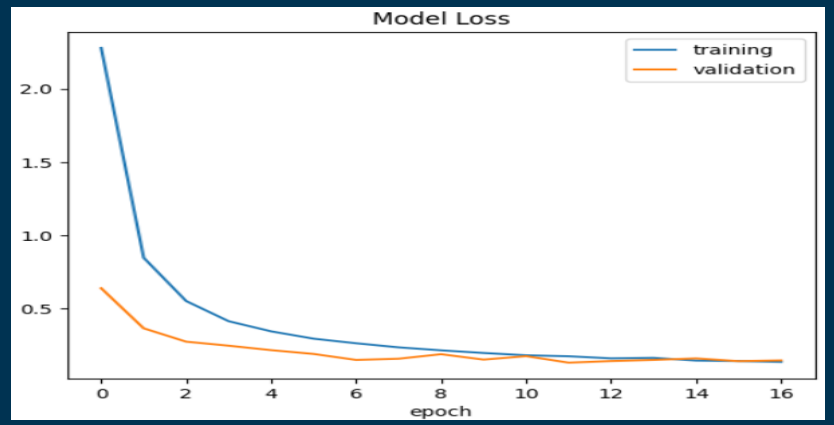
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 60)	1560
conv2d_1 (Conv2D)	(None, 24, 24, 60)	90060
max_pooling2d (MaxPooling2D)	(None, 12, 12, 60)	0
conv2d_2 (Conv2D)	(None, 10, 10, 30)	16230
conv2d_3 (Conv2D)	(None, 8, 8, 30)	8130
max_pooling2d_1 (MaxPooling2D)	(None, 4, 4, 30)	0
flatten (Flatten)	(None, 480)	0
dense (Dense)	(None, 500)	240500
dropout (Dropout)	(None, 500)	0
dense_1 (Dense)	(None, 43)	21543

Total params: 378,023
Trainable params: 378,023
Non-trainable params: 0

We have used EarlyStopping callback to prevent overfitting by stopping the training process early if the validation loss does not improve for a certain number of epochs (specified by the patience parameter).

Results



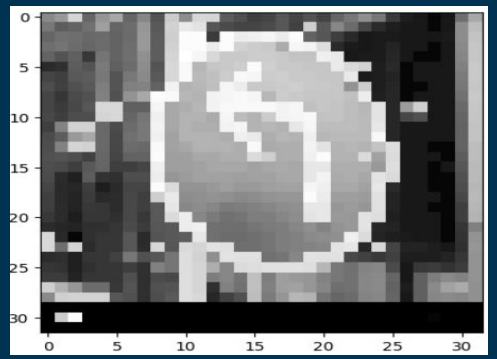


The Accuracy score is: 0.9543151259422302

Time to test the image on real images from the internet

Image

Preprocessed Image



```
[ ]  predict_x=model.predict(img_test_1)

     print("The Predicted sign: "+ str(np.argmax(predict_x,axis=1)))

     1/1 [==============================] - 0s 216ms/step
     The Predicted sign: [34]

[ ]  print(data["SignName"][34])

     Turn left ahead
```

So , The model passes the test

Conclusion : A convolutional neural network model was trained and implemented to classify traffic signs, and the results show that this technique outperforms the traditional models in terms of efficiency, accuracy, low losses, and execution time. The accuracy was estimated to be around 95.4%