An-Najah National University

Faculty of Engineering and IT

Department of Telecommunication Engineering

Graduation Project 1

Traffic sign recognition by using convolutional neural network

Supervisor: Dr. Allam mousa

Proposed by:

Sondos Khatatbeh [11821330]

Salam Nazzal [11819640]

Academic year: 2021/2022

Contents

List of figures

List of abbreviations

ASI: Artificial Super Intelligence

ML: Machine Learning

DL: Deep Learning

ADAS: Advanced Driver Assistance System

TSR: Traffic sign recognition

HOG: Host-Of-Orientated gradient

SVM: Support Vector Machine

GTSRB: German Traffic Sign Recognition Benchmark

VGG: Visual Geometry Group

BN: Batch Normalization

GAP: Global Average Pooling

NN: Neural Network

CNN: Convolutional Neural Network

ReLU: Rectified Linear Unit

SGD: Stochastic Gradient Descent

AdaGrad: Adaptive Gradients

Adadelta: Adaptive delta

RMSProp: Root Mean Squared Propagation

Adam: Adaptive moment

RNN: Recurrent Neural Network

RBFNN: Radial Basis Function Neural Network

FNN: Feed Forward Neural Network

GUI: Graphical User Interface

Abstract

In recent years there has been a rapid increase in technology that brought changes in human's life which helps us to make tasks so easier even the complex management systems. traffic sign recognition is one of the important factor to be considered. To recognize the traffic signs we build a model using convolutional neural networks and this model will recognize the traffic signs. This algorithm is optimized by different optimizers' algorithm such as Adam, AdaDelta, AdaGrad, RMSprop and SGD. The SGD algorithm achieved most accuracy so we consider it for optimizing our CNN model. It is implemented by using python platform.

Acknowledgement

Initially we extend our thanks and appreciation to Dr. Supervisor of the Project, Allam mousa and we extend our sincere thanks to everyone who helped us to complete the first part of this project We extend my sincere thanks to all the teachers represented by the department head Dr. Yousuf dama.

Chapter one

Introduction

1.1 Artificial intelligence

Artificial intelligence is the ability of machines to display similar intelligence to human. There are three main types of artificial intelligence: Artificial narrow intelligence (ANI) is weak or narrow AI because it displays intelligence less than the human brain, it cannot do several tasks or make intelligent decisions without human intervention, artificial general intelligence (AGI) demonstrates intelligence equal to humans, it is able to do any task that a human can, and it can solve multiple problems rather than a single one, artificial super intelligence (ASI) demonstrates intelligence exceed human intelligence so it become self-improving.

1.2 Machine learning

Machine learning is an AI subfield that involves teaching a machine how to make accurate predictions when given data. It is divided into two types: supervised and unsupervised learning. Systems are subjected to a wide amount of labeled data in supervised learning. Before we begin training, we must first decide which data to collect and which data features are important, after which the data will be prepared for processes such as normalization, deduplication and error correction. The next step will be to select a suitable machine-learning model from the numerous options available, each with its own set of advantages and disadvantages depending on the type of data. The data collected is then divided into two parts: training and evaluation. This evaluation data allows the trained model to be tested, to see how well it is likely to perform on real-world data. So the training data builds up the machine learning algorithm. The data scientist feeds the algorithm input data, which corresponds to an expected output. The model evaluates the data repeatedly to learn more about the data's behavior and then adjusts itself to serve its intended purpose, during training, validation data infuses new data into the model that it hasn't evaluated before. Validation data provides the first test against unseen data, allowing data scientists to evaluate how well the model makes predictions based on the new data. Not all data scientists use validation data, but it can provide some helpful information to optimize hyper parameters, which influence how the model assesses data. After the model is built, testing data once again validates that it can make accurate predictions. If training and validation data include labels to monitor performance metrics of the model, the testing data should be unlabeled. Test data provides a final, real-world check of an unseen dataset to confirm that the ML algorithm was trained effectively. In unsupervised learning, the data is unlabeled, so it identify patterns in data, trying to spot similarities that split that data into categories.

1.3 Deep learning

Deep learning is a subfield of machine learning that is essentially a three-layer neural network. These neural networks attempt to model the human brain's behavior in order to learn from large amounts of data. While a single-layer neural network can still make approximate predictions, adding hidden layers can help to improve accuracy. Neural networks are interconnected layers of algorithms, called neurons that feed data into each other, with the output of the preceding layer being the input of the subsequent layer. Each layer can be thought of as recognizing different features of the overall data, with the output of the preceding layer being the input of the subsequent layer. During the training process, the network gradually adjusts the importance of data as it flows between the layers of the network, learning how to recognize the pixels. This is possible because each link between layers has a weight associated with it, which can be increased or decreased to change the significance of that link. The system will check whether the neural network's final output is getting closer or further away from the desired output at the end of each training cycle. The system will then work backwards through the neural network, changing the weights attached to all of these links between layers, as well as a value called bias, to close the gap between the actual output and the desired output. Back-propagation describes this process. Finally, this process will settle on weights and bias values that will allow the network to accurately perform a given task, and the network will be said to have "learned" how to do a specific task.

1.4 An overview of this project

There are different types of algorithms in deep learning that we want to discuss. Every algorithm is designed to solve type of problems, we want to choose the suitable one for our project to build the model by it. We want to recognize the traffic signs and classify it by using convolutional neural network and understand how the training and validation are done by it. Before we implement this project by CNN algorithm, we want to understand about the neural network and how the CNN differs from ordinary neural network. So this report is organized as follows: in the first chapter we give an introduction of the project, in the second chapter we present the previous related works of the project, in chapter three we talk about the methodology of the project, chapter four gives the result and analysis of the implementation of the project, chapter five gives the conclusion and future work and the final chapter gives the references.

Chapter two

Literature review

Advanced Driver Assistance Systems (ADAS) rely heavily on traffic sign recognition (TSR) (ADAS). Traffic signs improve traffic safety by notifying drivers of speed limits and potential hazards . At 2009, in(1) train the histogram-of-oriented-gradient (HOG) descriptors of each class using one-versus-all SVMs. At 2010 In (2), Support vector machines are used. A staggering 36,000 Spanish traffic sign samples are included in the collection. 193 different sign classes with accuracy 95.5%. At 2012 in (3),Using different-sized histogram-of-oriented-gradient (HOG) descriptors and distance transforms, compare the performance of k-d trees, random forests, and support vector machines (SVMs) for traffic-sign categorization (DTs). To lower memory requirements and improve performance, we additionally use Fisher's criterion and random forests for feature selection. The data set used is the German Traffic Sign Recognition Benchmark (GTSRB), which includes 43 classes and over 50 000 images. At 2013 in (4), offer a hierarchical classification system for traffic signs. The approach has two hierarchies: the first divides traffic signs into numerous super classes, while the second divides the signs further within their super classes and outputs the final findings. Before the second hierarchy, two perspective correction strategies are developed and implemented, considerably improving classification accuracy. Experiments reveal that the proposed method exceeds the state-of-the-art method on the German Traffic Sign Recognition Benchmark (GTSRB), with an accuracy of 99.52 percent. Furthermore, processing one image takes roughly 40 milliseconds, making it suited for real-time applications. At 2015 in (5), For sign classification, a nearest neighbor classifier is used. The SURF algorithm extracts the training characteristics.

At 2017 in (6), The convolutional neural network (CNN) has been widely employed in high-accuracy traffic sign identification. In this research, we use a combined CNNs (CCNN) to modify hierarchical traffic signs, where the probabilities of superclass and subclass the sign belongs to are determined using two CNNs with a simple network.

At 2018 in (7), icronNet is a deep convolutional neural network that is extremely small. MicronNet's total architecture is thus constructed with as few parameters and calculations as possible while preserving recognition performance, resulting in the proposed network's optimal information density. The resulting MicronNet has a model size of just 1 MB with 5 10000 parameters (27 fewer than the state-of-the-art) but still attaining top-1 accuracy of 98.9% on the German traffic sign recognition benchmark.

At 2021 in(8) ,Because of the fast execution and high recognition rate of convolutional neural networks (CNNs), many traffic sign recognition algorithms have been presented. When compared to existing techniques, the suggested method creates an upgraded VGG convolutional neural network with much better performance. To further improve the overall architecture and accelerate calculation, some unnecessary convolutional layers are efficiently removed from the VGG-16 network, and the number of

parameters is considerably reduced. To improve accuracy without increasing the number of parameters, the BN (batch normalization) and GAP (global average pooling) layers are added to the network. When employing the enhanced VGG-16 network, the proposed technique only requires 1.15 M.

Chapter three

Methodology

3.1 Standards and specifications

We used the ISO/IEC as a standard in this project, it shows the input data for machine learning, the artificial neural network that contain of neurons connected of weighted links and non-linear functions as a threshold function, sigmoid function and polynomial functions, the testing data that input data when we want to generalize the final machine learning model, training data that used to fit the model and validation data to evaluate the prediction error during training.

3.2 Neural network

Neural network is computational model has a network architecture that made of artificial neurons. This structure has specific parameters that can modify it for performing certain tasks. Neural network contains of these layers: input, hidden and output layer. Single neural network is network without any hidden layer, multi-layer neural network is either shallow with single hidden layer or deep neural network with multi hidden layers. These layers contain of neurons which is mathematical functions, each input number in input layer have weight that is randomly initialized by the model, these weights are what makes each neuron unique. They are fixed during testing, but during training these are the numbers we're going to change in order to 'tune' our network. The inputs are multiplied by corresponding weights and sum everything together, the result of that is a number then it is inputted to nonlinear activation function, it is function that is applied to see if the nodes active or not and enable the CNN model to learn more complex things. The neural network without activation function is just a linear regression model that means the weights and bias would simply do a linear transformation. The nodes is connected to each other that means the output of each layer in neural network is become input of the next layer.

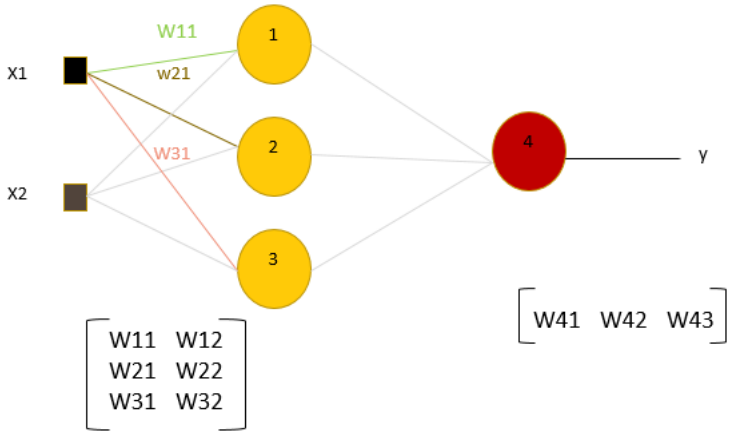


Figure 1: example of simple neural network with weights

$v = xw + b$ …….1

V is the number before inputted to activation function, x is the input matrix, w is the weight and b is the bias.

If the activation function is ReLU:

$\varphi(v) = ReLU = \max(0, v)$ …….2

There is many activation functions that we can use in neural network, and the most commonly used is ReLU, softmax

- **ReLU (rectified linear unit):**
  Equation is f(x) =max (0, x), derivative is f'(x) = {1; if v>0, 0; if v<0 and undefined if v=0} and the range is (0, +∞). This function is used in hidden layers

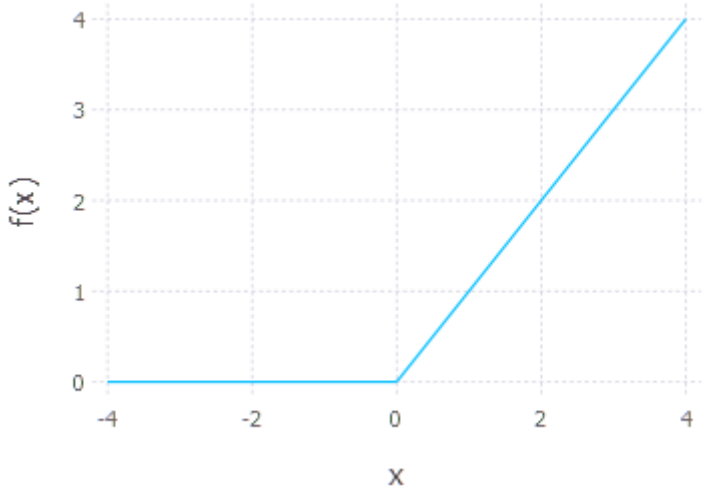


Figure 2:ReLU function

- **Softmax:**
  Equation is $pi = \frac{e^{ai}}{\sum_{k=1}^{N} e^{ak}}$ where pi the probability of each output class, N is the number of neurons in the output layer, ai is each output from the previous layer in the network and the range is (0, 1). It is used only for the output layer, for neural networks that need to classify inputs into multiple categories.

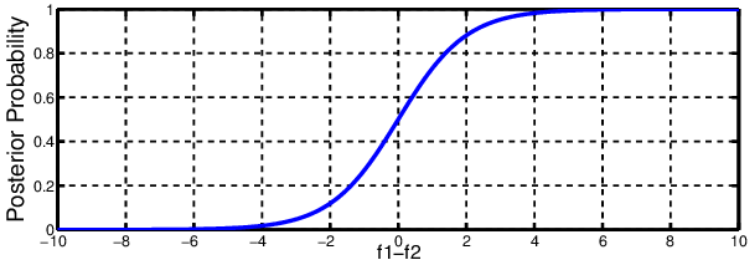


Figure 3:softmax function

Before we talk about training and testing in neural network we should know about loss function, the loss function uses two parameters to calculate the error, the first parameter is the estimate output (prediction) and the second one is the actual output (label). The prediction error tells the network how off their prediction from the actual output and then this error will be optimized during the learning process. There are different types of loss functions used for different types of problems but in most cases we use a cross entropy loss on the output.

- **Cross entropy loss**
  Is widely used to measure the performance of model. As we know that the output from the model is probabilities of each category that calculated from softmax function. The cross entropy loss is:

$$H(p, y) = -\sum yi \log(pi)$$

  Where y the desired output $i \in [1, N]$ N is the number of classes and pi is the probability from softmax function.

When we start off with neural network, the weights initialized randomly. Obviously, it won't give very good results. In the process of training, we want to start with a bad performing neural network and end with network with high accuracy. In terms of loss function, we want our loss function to much lower in the end of training. To optimize this loss we can use many optimization functions but the most commonly used is SGD because of its good results for optimization.

Optimizers is methods or algorithms used to change the attributes (weights and learning rate) of neural network to reduce the losses that computed by loss functions. Weights in neural network

decide how much influence the input will have on the output and the learning rate represent how fast the algorithm learns. There are different types of optimizers such as: Gradient Descent, Stochastic Gradient Descent, adagrad, adadelta, rmsprop and adam.

3.3 An overview of the CNN algorithm

The convolutional neural network is a deep learning algorithm that commonly used for computer vision applications. It is best suited for image classification and is used to classify images into different categories to which they belong. It has been observed that CNN is more efficient and faster than a regular deep neural network for problems related to computer vision. A Convolutional Neural Network are very similar to ordinary neural networks. They are made up of neurons with learnable weights and biases which is also known as supervised learning. All the basic idea learned or used for ordinary neural networks are also applicable to CNN's. The only difference between CNN and the ordinary neural network is that CNN assumes that input is an image rather than a vector. This vastly reduces the number of parameters to be tuned in a model. Convolutional neural networks or CNN's are very important in the computer vision field. CNN help in running neural networks directly on images and are more accurate than many of the deep neural networks.

Convolutional Nets models are easy and faster to train on images comparatively to the traditional models

3.2.1 Network layers

3.2.1.1 Convolutional layer

In this layer the input image (N-dimensional metrics) convolved with a set of convolutional kernels (filters) to generate the feature map as output of this layer.

But what is the kernels?

The kernel is matrix that moves over the input image (matrix of pixels), performs the dot product between the sub-region of the input data and the kernel to get the output as matrix of dot products. It moves over the input data by stride value (if the stride value is 1 so the kernel moves by one column of pixel in the input matrix), it is used to extract the high level features from the image.

There are different types of filters to extract the features like Gaussian blur, Gabor or prewitt filter for edge detection. We can choose the size of filters as we want but the most appropriate one is 3*3.

The values of these filters are learned automatically by the neural network through the training process, and the filters kernels which results in the features that are most efficient for the particular classification or the detection are automatically learned.

The convolution operation

The input image is multi channeled image as RGB image (3 channels) or single channeled as gray-scale image (1 channel) so the filter that we have to choose is either 2d (if gray scale image) or 3d (if multi channeled image). In convolution operation, we take the filter and slide it over the input image horizontally and we take the dot product between kernel and input image by multiplying the corresponding values of them and sum up all values to generate one value in the output feature map, this process continues until the kernel can no longer slide further.

3.2.1.2 Pooling layer

The output of convolutional layer become the input of this layer, so the feature map is sub-sampled by this layer, it takes the larger size feature maps and shrinks them to lower sized feature maps. While shrinking it always preserve the most dominant features in each pool steps.

There are different types of pooling techniques are used in different pooling layers such as max pooling, min pooling, average pooling, but the most popular of them is max pooling.

3.2.1.3 Activation function

In CNN architecture, after each learnable layers (layers with weights) nonlinear activation layers are used, this non-linearity behavior enables the CNN model to learn more complex things. The most commonly used activation functions in CNN is (ReLU, softmax)

3.2.1.4 Fully connected layer

The output of the pooling layer is set of the feature maps, those metrics are flattened to create a vector and this vector become the input of this layer to generate the final output of CNN, this layer is used for classification.

Fully connected layers is a multilayer perception contains of 3 types of layers: input, hidden and output layer.

- The input layer receives the features generated by convolution and pooling layers.
- The hidden layer is a sequence of neurons with weights that will be learned in training step
- The output layer is also contain of sequence of neurons, it has a different activation function. The softmax activation function is usually used to generate the probabilities of each category.

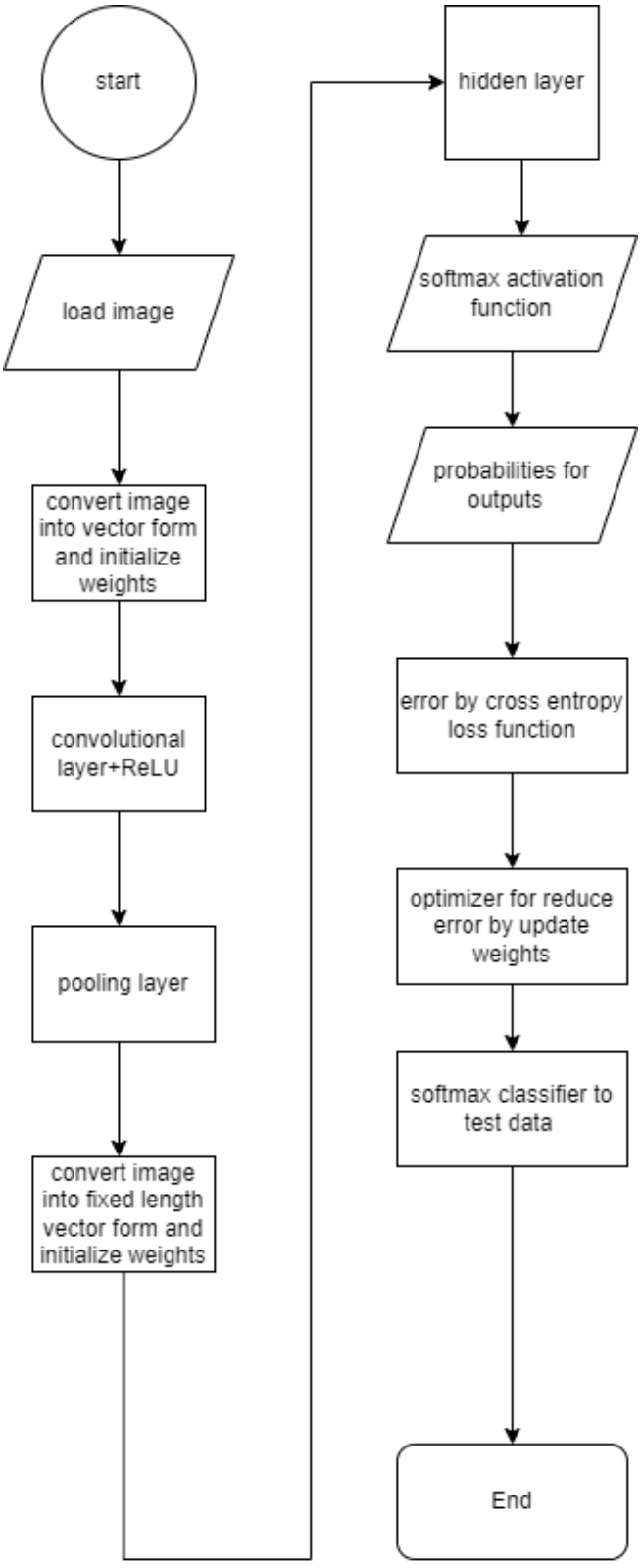


Figure 4: flowchart for CNN algorithm

3.4 Traffic sign recognition

After the traffic sign has been detected, the recognition and classification of it is done by using convolutional neural network algorithm.

The implementation of traffic sign recognition

The dataset that we use is GTSR, it contains more than 50,000 images in total and more than 40 classes, the training data contains of 39,209 images existing in 43 classes, each class represents a specific sign. The testing data is almost 12630 images.

The number of images in each class is shown in figure:

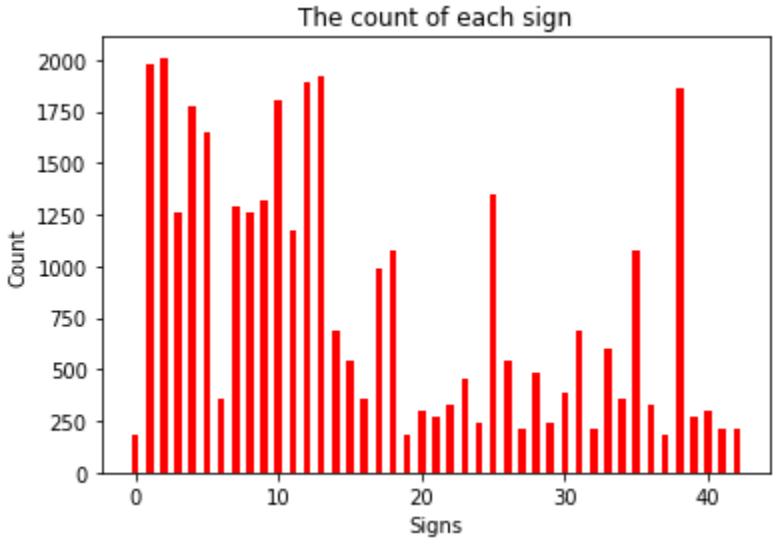


Figure 5: number of images in each class

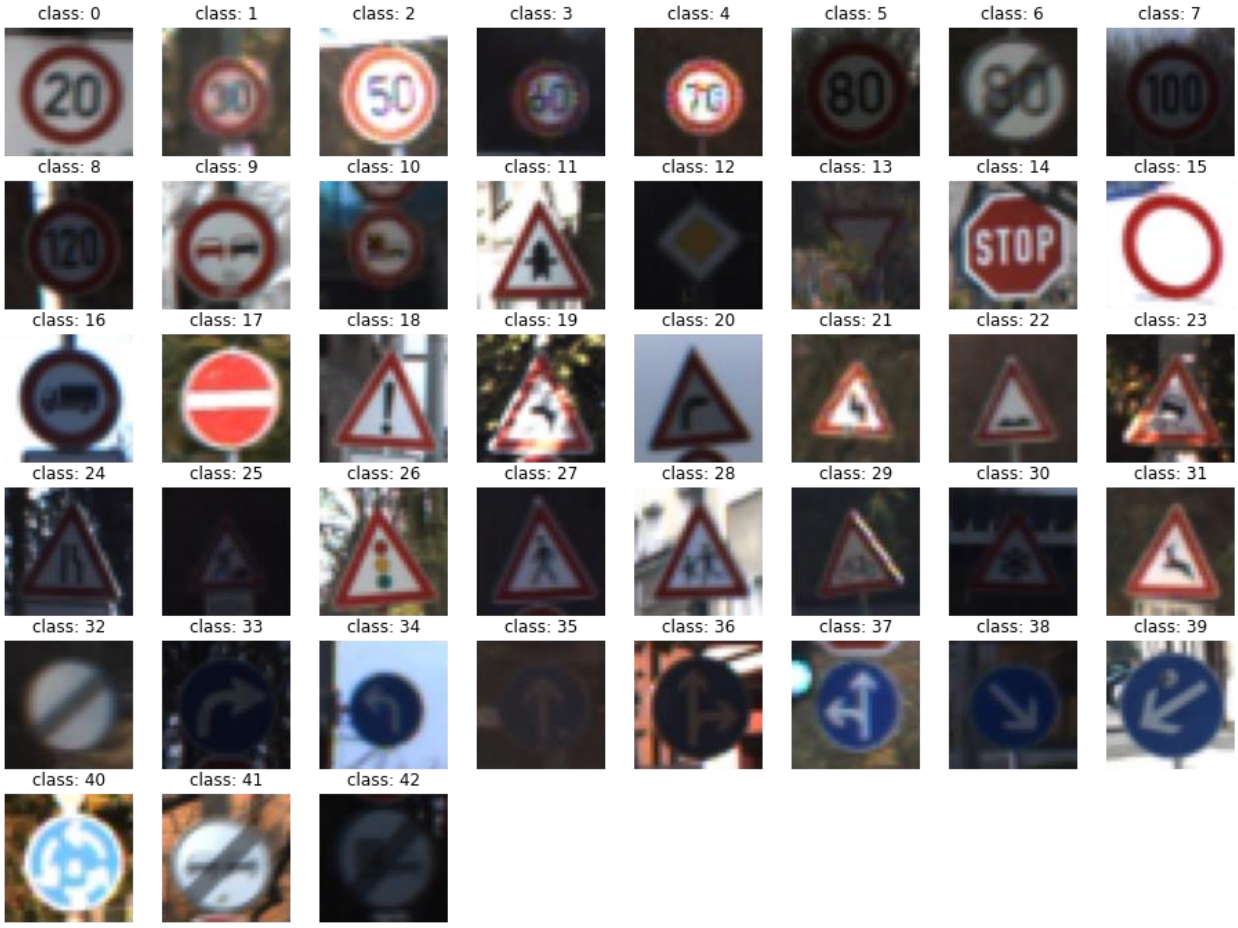


Figure 6: some traffic sign images in Dataset

3.4.1 The first model

In our model the size of input image is 30*30 and we use valid padding (the size of image is shrinking because p=0), in pooling layer the pool size is 2*2

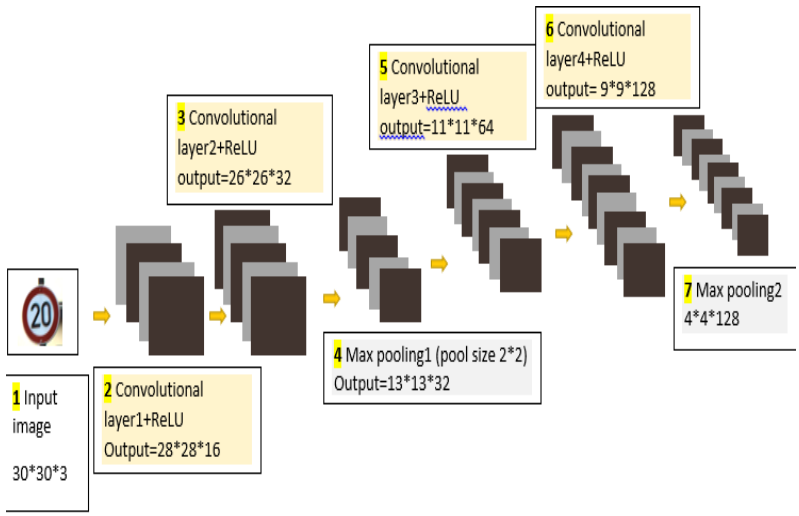


Figure 7: feature extraction network

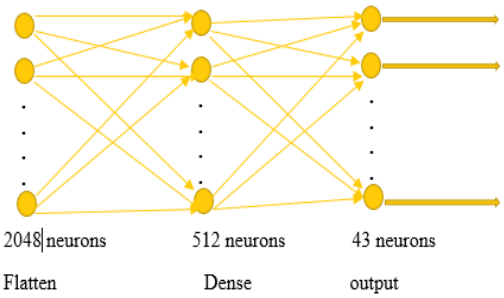


Figure 8: classifier network

The filters in CNN is trainable, that means the model is learning it by training. So these filters start to extract the highly component features in the image to recognize the traffic sign signal. The filters and feature maps after each layer are shown as follows:

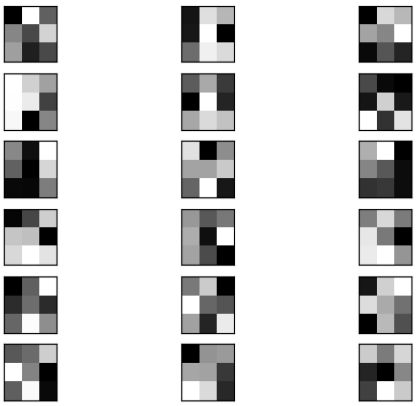


Figure 9: some filters in conv1

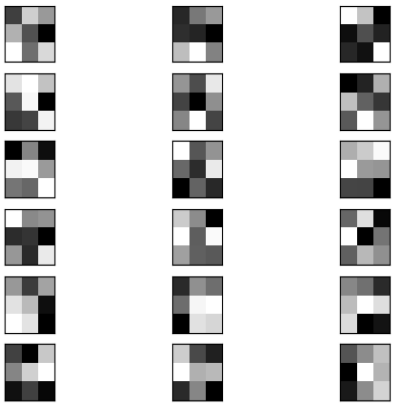


Figure 10: some filters in conv2

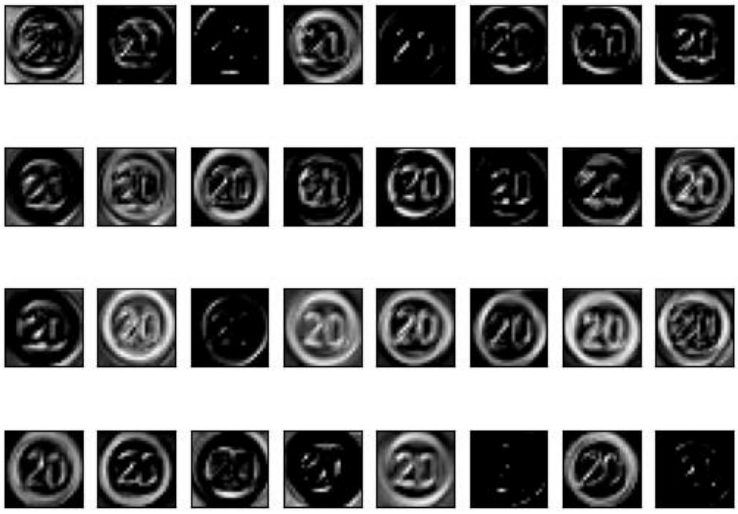


Figure 11: some features map after conv1

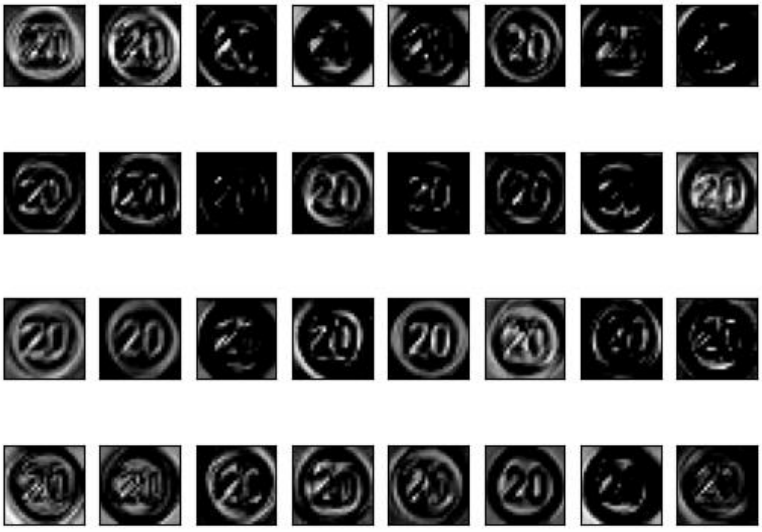


Figure 12: some features map after conv2

Figure 13: some features map after conv3

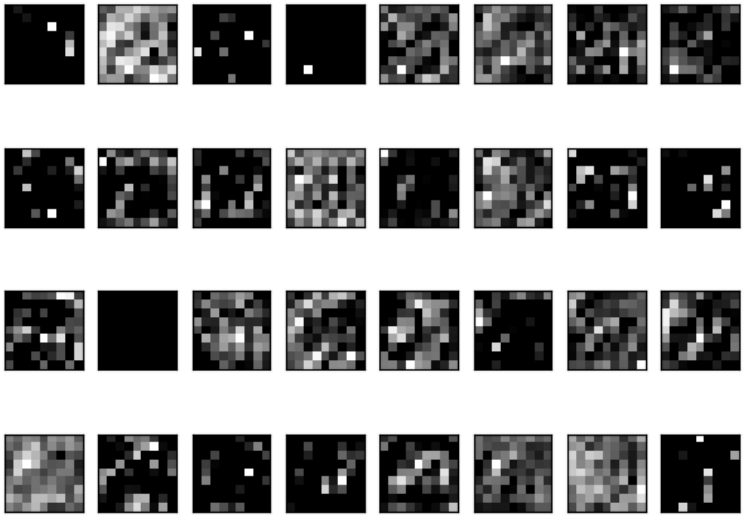


Figure 14: some features map after conv4

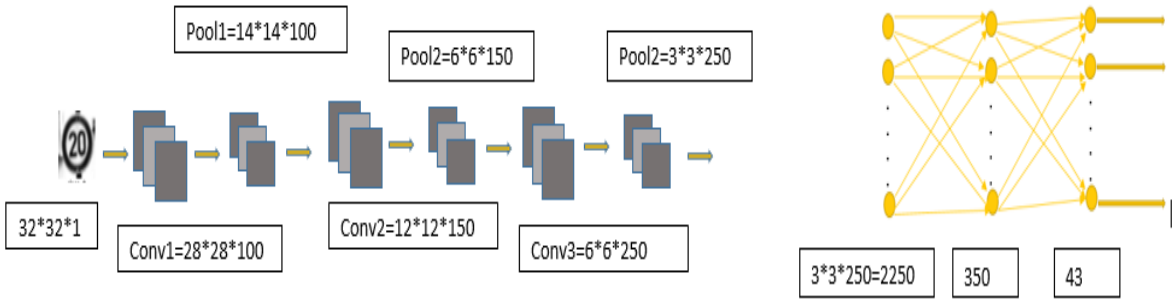
Feature map size=$1+\frac{w-f+2p}{s}$

W: input size= 30, f: filter size=3, s: stride=1, p: padding=0

3.4.2 The second model

There is a main problem appears in overfitting that reduces the model's ability to generalize, so we used techniques to prevent it, dropout and batch normalization. We modified the old model in CNN layers as shown:

In our new model the size of input image is 32*32 and we use valid padding (the size of image is shrinking because p=0), in pooling layer the pool size is 2*2



To make it easier for the neural network to analyze the input images by removing any unwanted biases, we convert them to a single-channel grayscale. This lowers the number of computations required because there are fewer channels after the conversion, which also increases model accuracy. The CNN model architecture is built as following steps:

1. The first convolutional layer contains of 100 filters with 5*5 filter's size so based to equation above the size of image becomes 28*28
2. Batch normalization layer for the first convolutional layer (conv1) in the network, the input to this layer is the output from the conv1 layer, which has 100 channels.
3. The first max pooling layer is 2*2 this means the pooling window will move 2 pixels at a time and take the maximum value of each 2x2 window. This will reduce the spatial dimension of the output by a factor of 2.
4. The second convolutional layer contains of 150 filters with 3*3 filter's size so based to equation above the size of image becomes 12*12.
5. Batch normalization layer for the second convolutional layer (conv2) in the network, the input to this layer is the output from the conv2 layer, which has 150 channels.
6. The second max pooling layer is 2*2 this means the pooling window will move 2 pixels at a time and take the maximum value of each 2x2 window. This will reduce the spatial dimension of the output by a factor of 2.
7. The third convolutional layer contains of 250 filters with 1*1 filter's size so based to equation above the size of image becomes 6*6.
8. Batch normalization layer for the third convolutional layer (conv3) in the network, the input to this layer is the output from the conv3 layer, which has 250 channels.

9. The third max pooling layer is 2*2 this means the pooling window will move 2 pixels at a time and take the maximum value of each 2x2 window. This will reduce the spatial dimension of the output by a factor of 2.
10. The fully connected layer contains of a linear layer in a PyTorch neural network, with 25033 input neurons and 350 output neurons. The Linear module applies a linear transformation to the input data, y = Wx + b, where W is the weight matrix, x is the input, and b is the bias. The number of neurons in the input layer is determined by the size of the input data (250 * 3 * 3) and the number of neurons in the output layer is determined by the argument passed in (350), it contains of another linear layer the input of it is 350 that the output of previous layer and the output of this layer is 43 neuron that the number of labels in traffic sign dataset.
11. Dropout layer creates a dropout layer in a neural network. The dropout layer is used to prevent overfitting by randomly setting a fraction of the input units to 0 at each update during training time, which helps to break the symmetry and make the model more robust. If (p=0.5) that determines the dropout rate, which is the fraction of the input units that will be set to 0 during training. In this case, it's set to 0.5, which means that half of the input units will be dropped out during training.

There are numerous issues with dataset which represent the real world problems faced in actual traffic sign recognition:

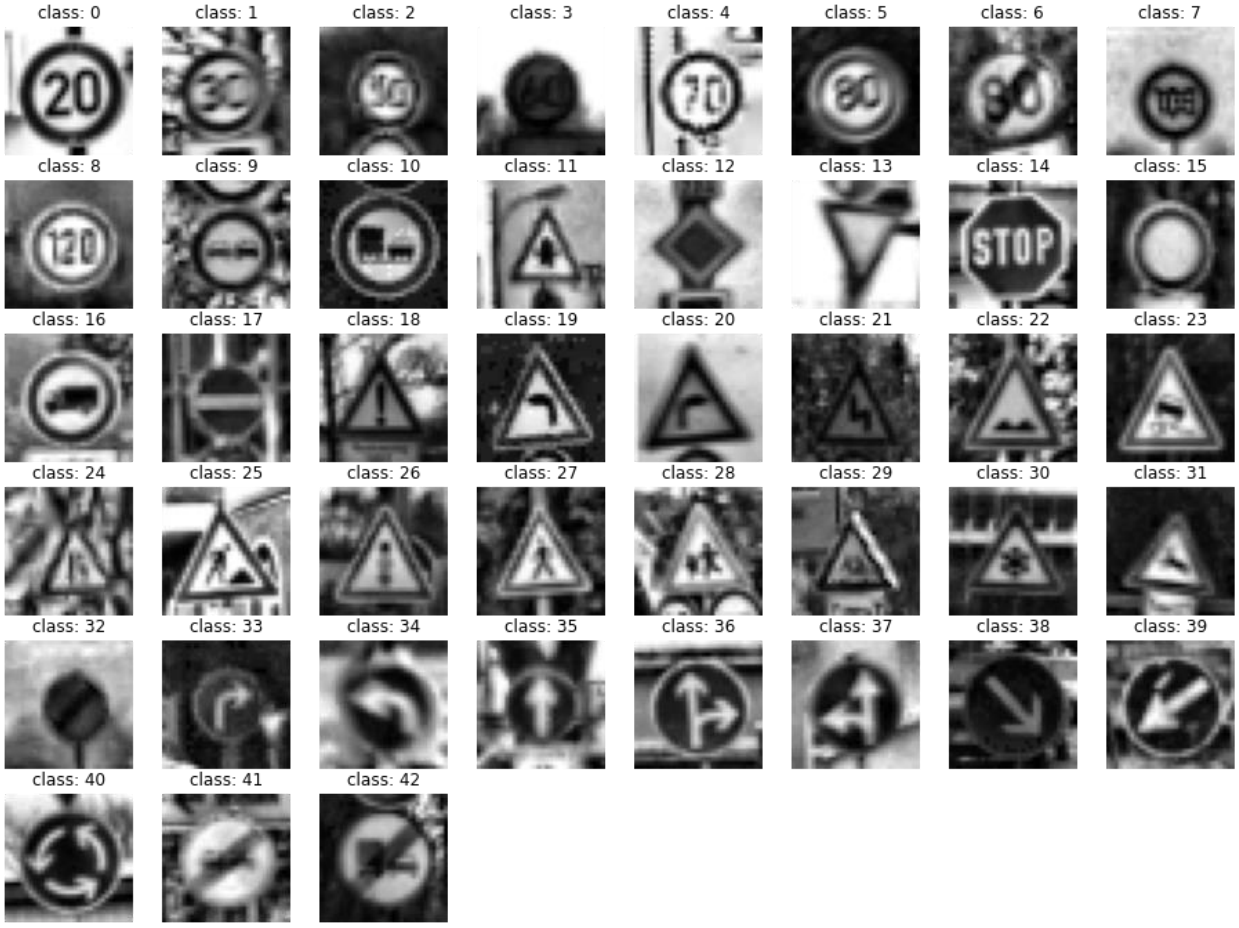
- Class imbalance: Apparently dataset is very unbalanced, some of the classes having 2000 samples and some of them having only 200 samples. This imbalance biases the model to predict classes with higher number of sample more frequently than the ones with less number of samples to achieve better accuracy.
- High contrast variation: The images differ significantly in terms of contrast and brightness. It is difficult to human to understand and classify some of these signs which are in total darkness.
- Small dataset: Even though we have about 35k train images, they are not quite enough to perform well in all general case scenario. More data also helps with overfitting.

So we try to solve these problems to generalize the model over new data and get good accuracy in testing and prediction. So we summarized this in three ways, CLAHE-gray for high contrast variation, flipping or augmentation for imbalanced dataset to increase the number of data in each class.

- CLAHE-gray:

CLAHE stands for "Contrast Limited Adaptive Histogram Equalization." It is a method for enhancing the contrast in images, specifically in areas that have low contrast. It works by dividing the image into small blocks called "tiles," and then applying histogram equalization to each tile separately. The contrast enhancement is limited by a parameter called "clip limit," which prevents

the amplification of noise or other small variations in the image. CLAHE is commonly used in image processing, particularly in medical imaging and microscopy.



- Flipping

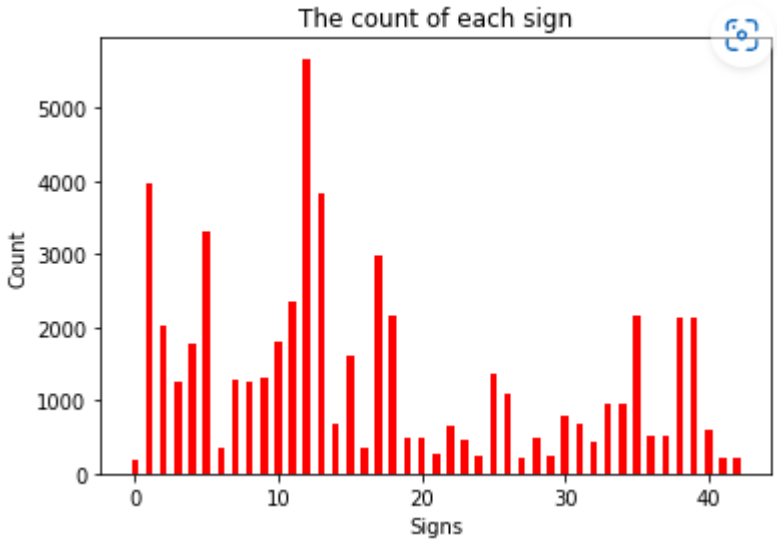
This technique helps to increase the size of the dataset and improve the robustness of the model by introducing new variations of the training data. We used horizontally, vertically, both and cross flipping.

Horizontally flipping: is a list of integers representing classes of traffic signs that can be flipped horizontally without affecting their meaning. For example, traffic signs with class number 11, 12, 13, 15, 17, 18, 22, 26, 30, 35 can be flipped horizontally.

Vertically flipping: is a list of integers representing classes of traffic signs that can be flipped vertically without affecting their meaning. For example, traffic signs with class number 1, 5, 12, 15, 17 can be flipped vertically.

Both flipping: is a list of integers representing classes of traffic signs that can be flipped horizontally and vertically without affecting their meaning. For example, traffic signs with class number 32, 40 can be flipped both horizontally and vertically.

Cross flipping: is a 2-dimensional array of integers representing classes of traffic signs that can be flipped horizontally and vertically and still retain their meaning. For example, traffic signs with class number 19, 20, 33, 34, 37, 36, 39, 38 can be flipped both horizontally and vertically.



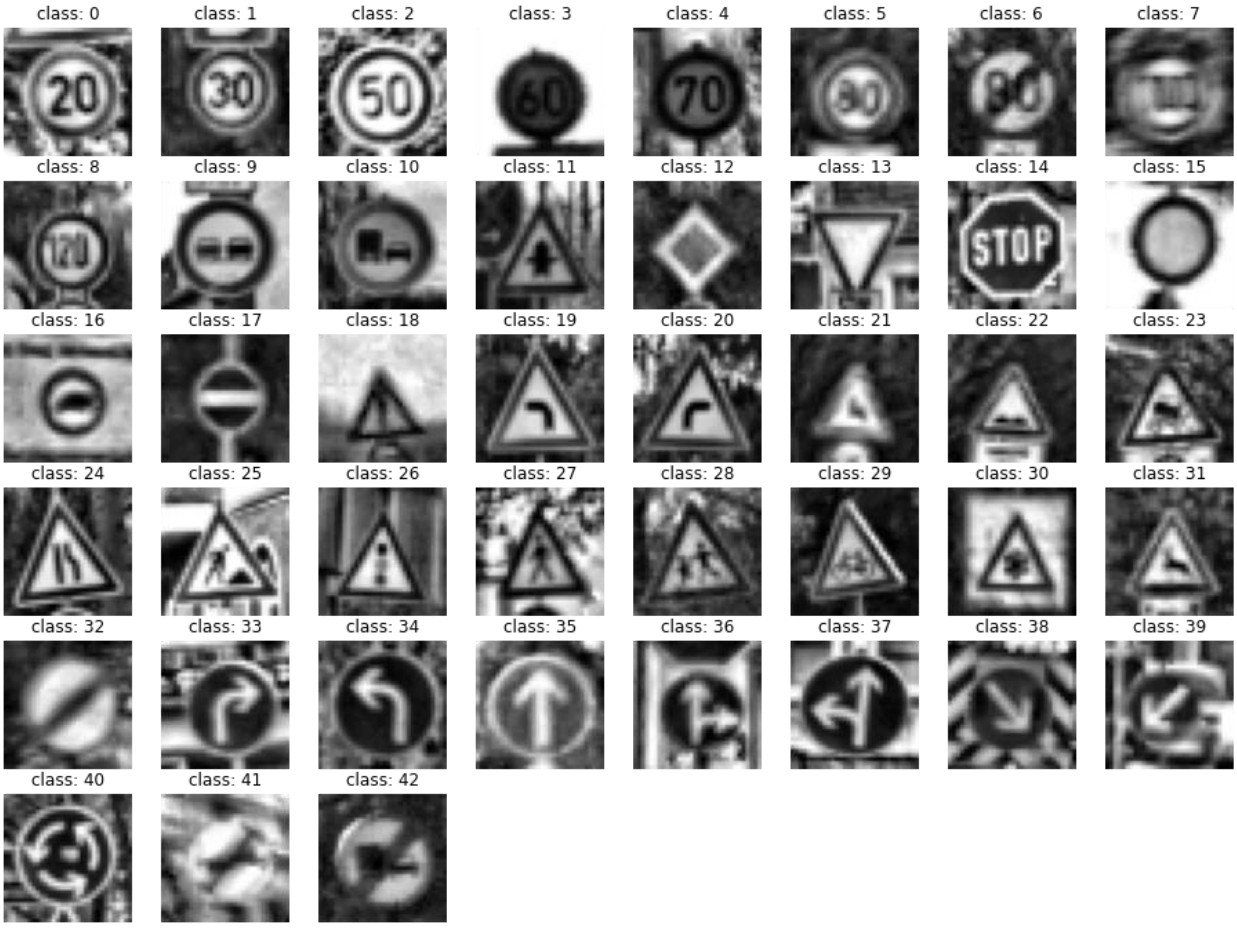
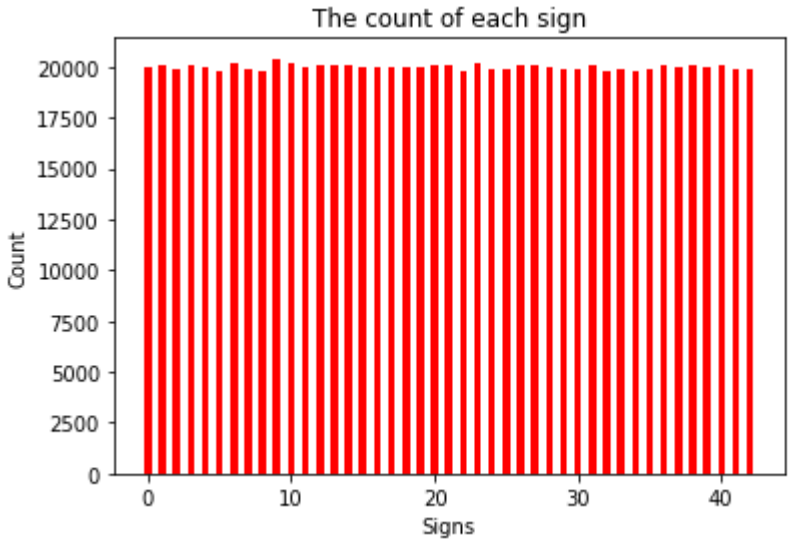


Figure 15:some images after flipping

- Augmentation

We want to get balanced dataset to improve the results. So, we built a dataset by geometrically transforming (translation, rotation, shear mapping, scaling) the same sign image. This can be done easily by applying torchvision.transforms in PyTorch

We used WeightedRandomSampler to sample from the dataset with the desired class balance. When the model is trained using this sampler, it will have a balanced exposure to all the classes, reducing the chances of overfitting to a specific class. It takes in two arguments, the first is the array of weights created earlier, and the second is the number of samples to generate from the dataset. In this case, it is 43 * 20000, where 43 is the total number of classes and 20000 is the number of samples to generate from the dataset. So classes with fewer samples will have higher weights and classes with more samples will have lower weights.



To implement this in python, we must detect the threshold that appears if the image is traffic sign or not. So we decided by experiments that the appropriate threshold in loading images: if max-probability more than 10, so it classified as traffic sign and if it less than 10 so it is an unknown class. Then we applied text to speech conversion in python code.

```
if maximum_prop > 10 : #  thershold 10%
    pred = torch.argmax(output, dim=1)

    sound=gtts.gTTS(classes_label[pred.item()],lang="en")
    engine.say(classes_label[pred.item()])
    engine.runAndWait()
```

Figure 16:some images after augmentation

Chapter four

Results and analysis

We use python platform to implement the CNN algorithm, so we load the GTSR dataset and install these packages: **Numpy** for math operation and adds support for matrices, **pandas** for make working with labeled data, **os** for managing the computer's memory and processes, **opencv** for all types of image analysis, **matplotlib** for various types of graphical representations, **pillow** for opening and saving images, **keras** for making the implementation of neural network easy, **tensorflow** for building and training neural network and **scikit-learn** for splitting the data into test and train.

Firstly we want to check the accuracy, it is for evaluating classification models and is fraction of predictions our model got right.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

We use GTSR dataset to recognize traffic sign images, the input is 30*30*3 pixel image (3channels because it is RGB image), the feature extraction contains: convolutional layer with 16 (3*3) filters and ReLU activation, another convolutional layer with 32 (3*3) filters and ReLU activation, the pooling layer employs the max pooling process of 2*2 submatrices, another two convolutional layers with 64 and 128 (3*3) filters and finally max pooling layer of 2*2 submatrices. The input of neural network is 7*7*128 (6,272) nodes, the hidden layer contains of 512 nodes with the ReLU activation and the output contains of 43 nodes with softmax activation to represent 43 classes (labels).

39,209 images are used for training with corresponding 39,209 labels, 30 epochs for training, 0.001 learning rate, batch size is 32.

When training the model, one of the main things that we want to avoid would be overfitting, this is when the model fits the training data well but it isn't able to make accurate predictions for new data it hasn't seen before. To find out if the model is overfitting, we split the training data into train and validation set, the training set is used to train the model while the validation set is used to evaluate the model's performance.

In these curves we want to show the accuracy and loss when using different optimizers for training set and validation set, when the two curves (accuracy or loss for training and validation set) are equal at an epoch this mean that the model is fitting the training set, it can predict on new data and avoid the overfitting.

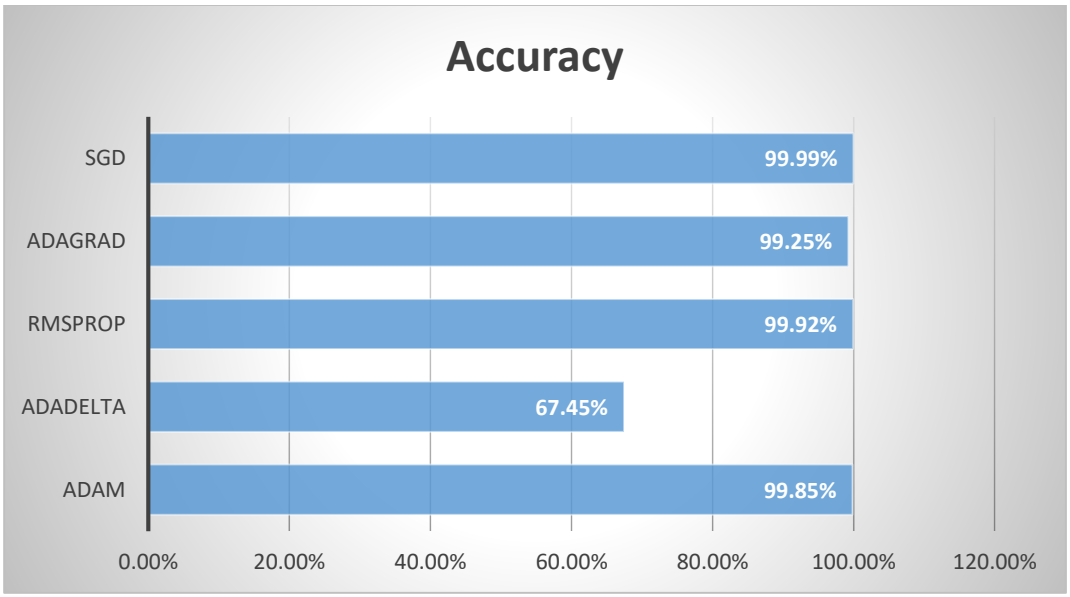


Figure 17: the accuracy of each optimizer

- **Accuracy and loss for SGD**

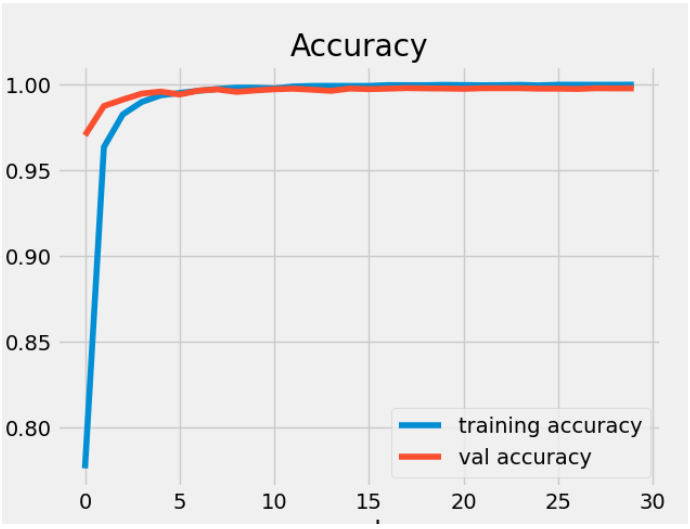


Figure 18: Accuracy for SGD optimization algorithm

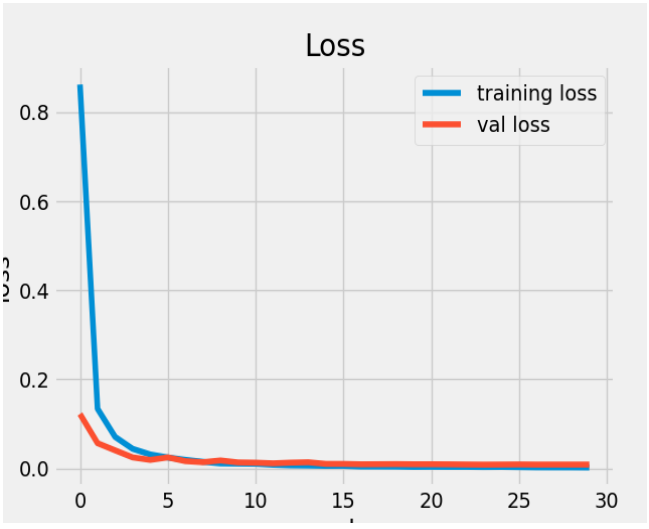


Figure 19: loss for SGD optimization algorithm

➢ **Accuracy and loss for Adam**

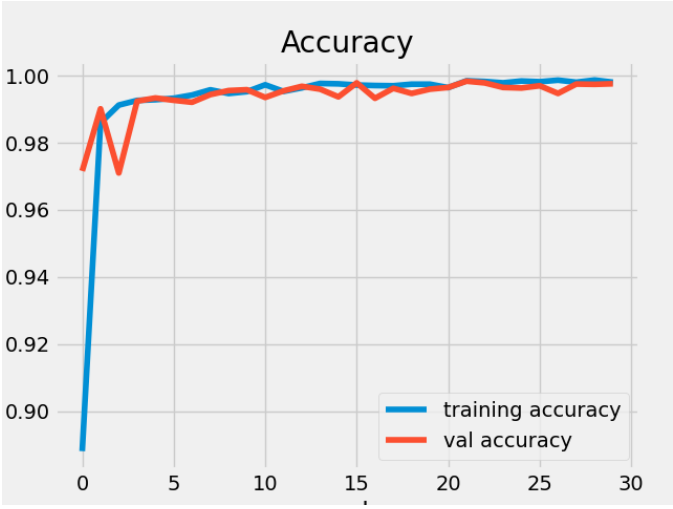


Figure 20: accuracy for adam optimization algorithm

Figure 21: loss for adam optimization algorithm

➢ **Accuracy and loss for Adadelta**

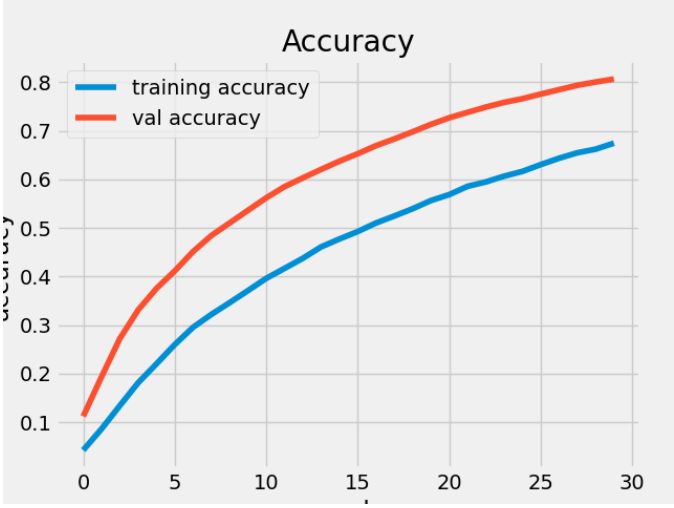


Figure 22: accuracy for adadelta optimization algorithm

Figure 23: loss for adadelta optimization algorithm

➢ **Accuracy and loss for Adagrad**

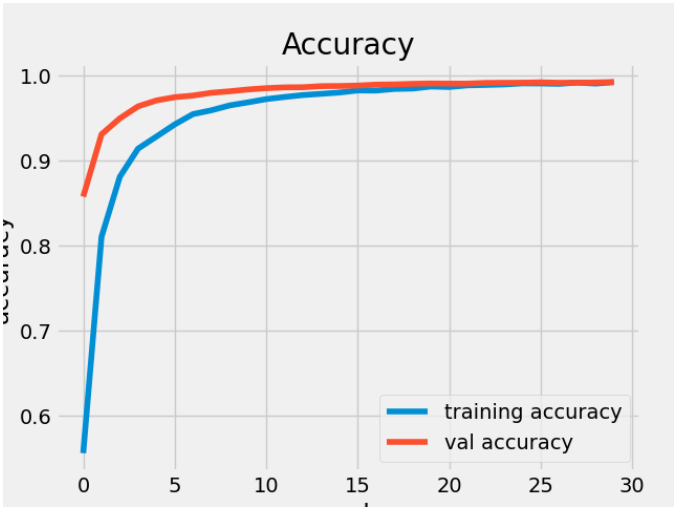


Figure 24: accuracy for adagrad optimization algorithm

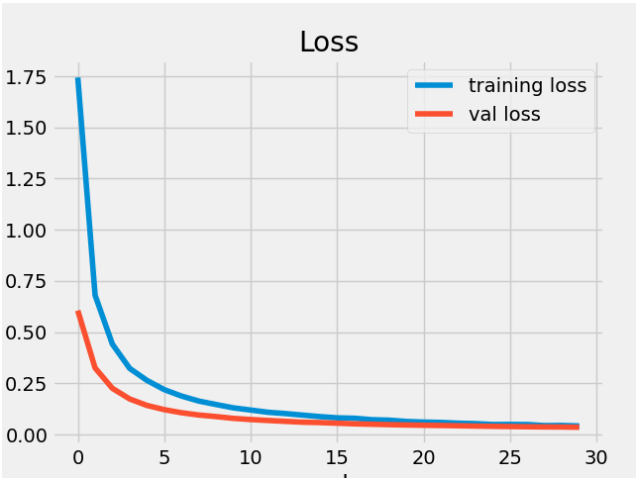


Figure 25: loss for adagrad optimization algorithm

➢ **Accuracy and loss for RMSprop**

+

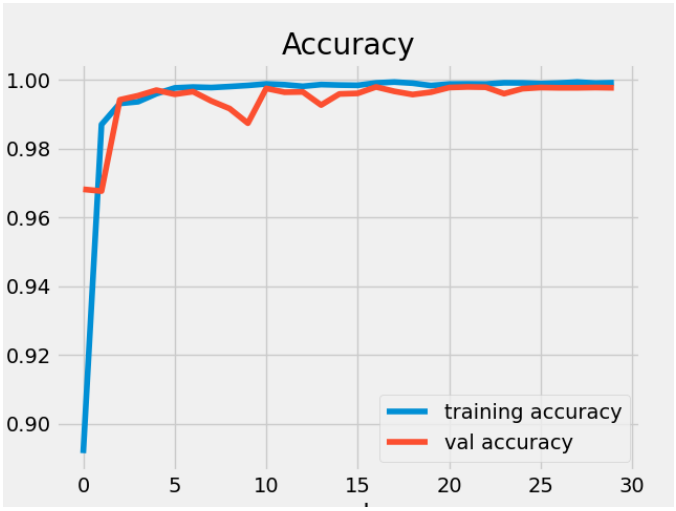


Figure 26: accuracy for rmsprop optimization algorithm

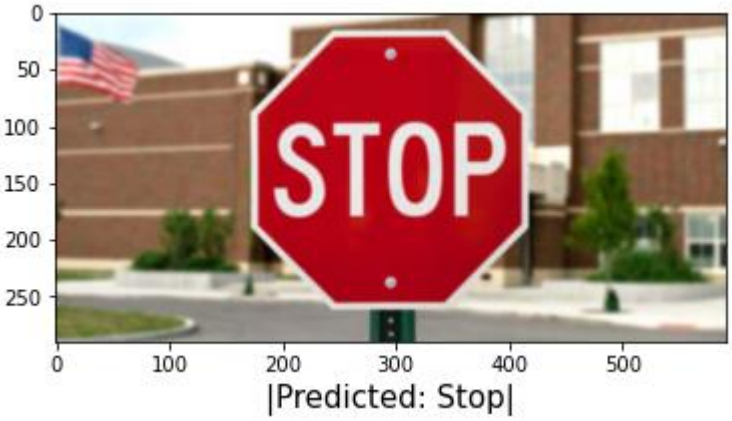
Figure 27: loss for rmsprop optimization algorithm

The result of test accuracy is shown in this table:

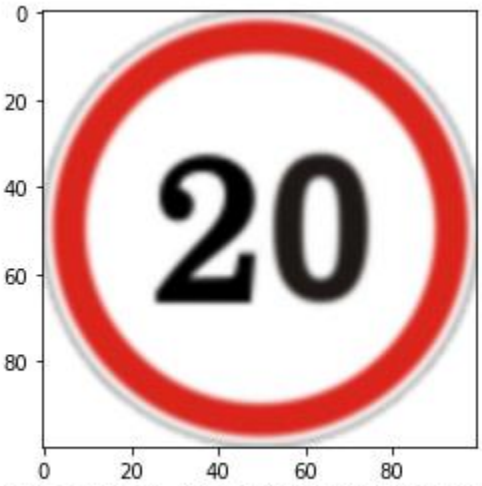
Model	Test accuracy*100%
The first model with RGB images	92.6%
The second model + change images to grey scale	95.23%
The second model + change images to grey scale + CLAHE	96.01%
The second model + change images to grey scale + CLAHE + Augmentation (final model)	99.446%

➢ Examples of predict some new images by final model

- **False and true predict of one new image example**

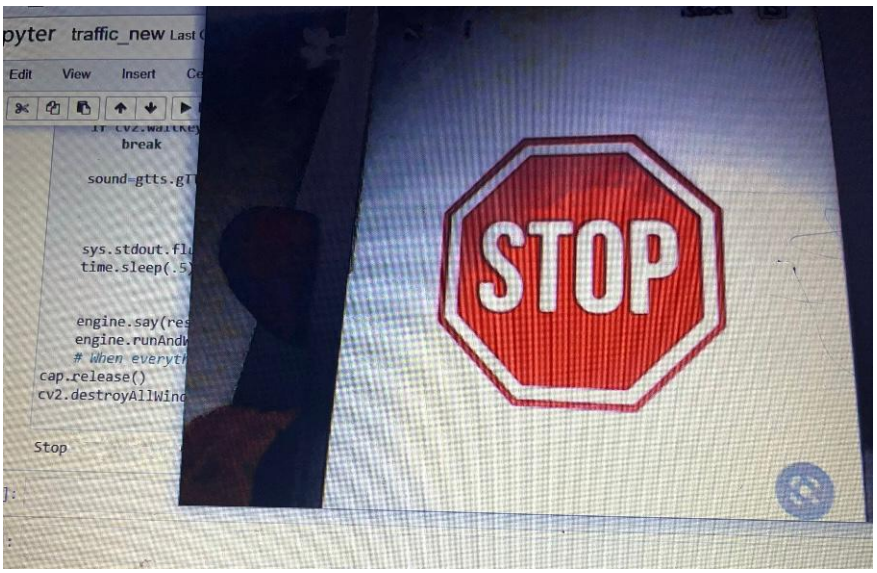


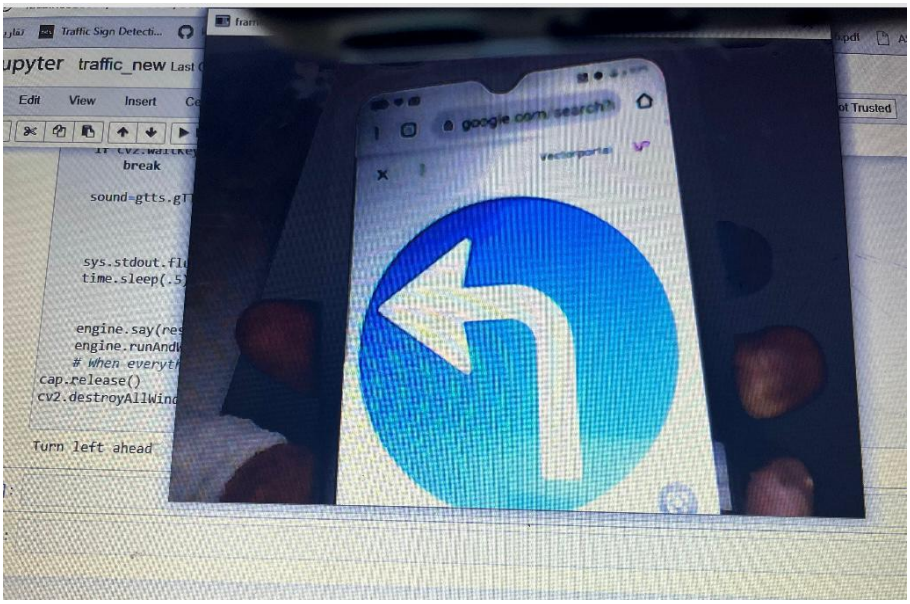
|Predicted: Stop|

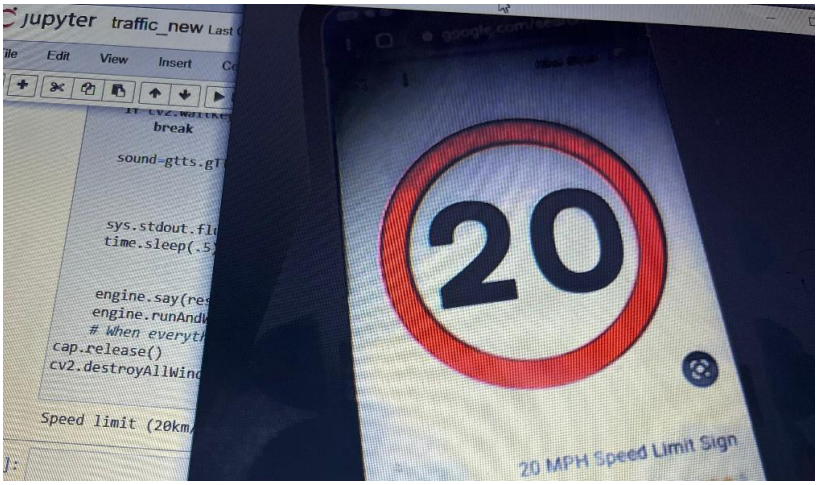


|Predicted: Speed limit (80km/h)|

- **Predict images using camera**







Chapter five

Discussion

We faced many problems in this project, the most important of it that the model can't do generalization, so the model trains well but it can't recognize new images. In CNN, to get a good model that can generalize new images we must prevent the overfitting and taking care of improving the test accuracy. So we modified the model and applying CLAHE and augmentation to get 99.446% test accuracy, this is very good accuracy to recognize new images.

This high accuracy doesn't mean that the model recognize all new images, it's still misrecognizing some new images, may the difference of some traffic signals in many countries cause this problem.

Another problems that we faced, the very huge time in training because of GPU in our device and the camera is very slow when we want to test some new images.

Chapter six

Conclusion and future work

In this project we discussed how the CNN algorithm trains the model to recognize and classify the traffic signs images by different optimization algorithms, and it is found that the SGD gives most accuracy than others. Trying new models to increase of test accuracy to do generalization very well and the final model achieved highest accuracy that is 99.446%. Camera of laptop and loading images is developed on python to test the traffic signs. We implemented this project to detect traffic signs on the road by using camera, recognize and classify them, and give the driver voice alarm containing the meaning of that signs by converting text to speech. This model can improved to detect signs in road by high quality cameras, recognize and classify them to help drivers and reduce accidents rate.

Chapter seven

References

(1) Y. Xie, L. Liu, C. Li, and Y. Qu, “Unifying visual saliency with HOG feature learning for traffic sign detection,” in Proc. IEEE Intell. Veh. Symp., 2009, pp. 24–29

(2) S. Maldonado Basc´on, J. Acevedo Rodr´ıguez, S. Lafuente Arroyo, A. Caballero, and F. L ´opez-Ferreras, “An optimization on pictogram identification for the road-sign recognition task using SVMs,” Computer Vision and Image Understanding, vol. 114, no. 3, pp. 373–383, 2010.

(3) Zaklouta, F.; Stanciulescu, B. (2012). Real-Time Traffic-Sign Recognition Using Tree Classifiers. IEEE Transactions on Intelligent Transportation Systems, 13(4), 1507–1514. doi:10.1109/TITS.2012.2225618.

(4) Wang, Gangyi; Ren, Guanghui; Wu, Zhilu; Zhao, Yaqin; Jiang, Lihui (2013). *[IEEE 2013 International Joint Conference on Neural Networks (IJCNN 2013 - Dallas) - Dallas, TX, USA (2013.08.4-2013.08.9)] The 2013 International Joint Conference on Neural Networks (IJCNN) - A hierarchical method for traffic sign classification with support vector machines. , (), 1–6.* doi:10.1109/IJCNN.2013.6706803

(5)Han, Yan; Virupakshappa, Kushal; Oruklu, Erdal (2015). *[IEEE 2015 IEEE International Conference on Electro/Information Technology (EIT) - Dekalb, IL, USA (2015.5.21-2015.5.23)] 2015 IEEE International Conference on Electro/Information Technology (EIT) - Robust traffic sign recognition with feature extraction and k-NN classification methods. , (), 484–488.* doi:10.1109/eit.2015.7293386

(6) Chen, Lingying; Zhao, Guanghui; Zhou, Junwei; Kuang, Li (2017). *[IEEE 2017 4th IAPR Asian Conference on Pattern Recognition (ACPR) - Nanjing, China (2017.11.26-2017.11.29)] 2017 4th IAPR Asian Conference on Pattern Recognition (ACPR) - Real-Time Traffic Sign Classification Using Combined Convolutional Neural Networks. , (), 399–404.* doi:10.1109/ACPR.2017.12

(7) Wong, Alexander; Shafiee, Mohammad Javad; Jules, Michael St. (2018). *MicronNet: A Highly Compact Deep Convolutional Neural Network Architecture for Real-time Embedded Traffic Sign Classification. IEEE Access, (), 1–1.* doi:10.1109/access.2018.2873948

(8) Bi, Zhongqin, et al. "Improved VGG model-based efficient traffic sign recognition for safe driving in 5G scenarios." *International Journal of Machine Learning and Cybernetics* 12.11 (2021): 3069-3080.