COMPUTER VISION I (CS 537)

Computer Vision Final Project Report

Name: Harsh Sandip Kothari **OSU ID: 934-062-578**

1) Problem statement:

a) State the problem:

Self-driving cars need traffic sign recognition to properly parse and understand the roadway. Even though there are many advancements in the field of object detection and recognition it remains to be a crucial challenge even today. Traffic signs are an integral part of our road infrastructure. They provide us with recommendations or suggestion that needs to be followed to avoid the accidents and without the road signs, the accidents will increase. So moving forward into the future where we want self-driving cars without any driver the traffic sign classification and recognition becomes an important task for self-driving cars as this will ensure safe transportation. So I will be creating a model that will be useful in recognizing and classifying the traffic signs into 43 different classes which in turn is useful in the future for autonomous driving.

b) Describe the scope, domain and challenges of your problem:

The main aim of the project to accurately classify the traffic signs and for that, I have used a dataset that contains 43 different classes of traffic signs which helps the model to be more precise and accurate about a traffic sign. Currently, the model that I will create is limited to just traffic sign classification but the model can also be used for other classifications such as number classification, shape classification, and a lot more. Since I will be working on the project individually all the tasks will be performed by me. To classify the signs, I am using the EdLeNet architecture which consists of 3 convolutional layers to improve the accuracy of the system and it is better than the simple LeNet architecture. For any given image, the model will be able to find the correct class it belongs to, and to make the output have accurate results the dataset consists of 43 different classes of traffic signs along the road, speed limit signs, yield signs, merge signs, etc. I have also used dropout which refers to dropping out units in a neural network and is used to reduce overfitting and improve the reliability. For this project, I will be using GTSRB - German Traffic Sign Recognition Benchmark dataset that consists of 43 traffic sign classes and nearly 50,000 images and the number of training images is 34799. The domain of my project is image recognition and classification and the model that I have created can be used for self-driving cars. The challenges with the selected dataset are that the images are of low resolution and the images are pixelated so many of the images are difficult to be recognized by the human eye and also the low light on roads at night so it becomes difficult to identify a sign. Another challenge is accuracy and I will be adding some features so that I can achieve an accuracy around 98%.

c) Importance of your problem:

Traffic signs are an integral part of our road infrastructure. Traffic sign recognition is just one of the problems that computer vision and deep learning can solve. This model will be useful in significantly improving safety and in the implementation of an important step on the way to autonomous driving. Being able to automatically recognize traffic signs enables us to build “smarter cars”. The same problem can be found in other fields where we need image recognition and some of the examples include lane recognition for smart cars, number recognition, and a lot more.

2) Approach:

a) Summary of the entire approach:

The dataset is first split in test, training and validation datasets, the training set consists of 34799 images, and the validation and test set consists of 4410 and 12630 images respectively. I have used Python with Keras to write the code. First, the images are converted into grayscale which helps in reducing the noise in the image if present. After that, the images are normalized and then they are being used to detect the key points and after that, the features are being detected which will be used keypoint matching with other images so that we can accurately classify them in the right class. The model that I used is inspired by the LeNet Architecture which is presented by Yann Le Cun. I have added some tweaks to that architecture to provide more accurate results and the name of the network is EdLeNet. The depth model doubles every layer. The network consists of 3 convolution layer and each of them is followed by max pooling. The network is trained using Adam optimizer. To improve the reliability of the model I have also used the dropout to improve the reliability and it also helps prevent overfitting. I have passed the validation data along with the training data to avoid overfitting as well. I have used a confusion matrix to evaluate the model. In the end, the accuracy that I was able to achieve on the training data was 99% and an accuracy of around 98% on the test data and 95% on the validation data.

b) **Overview figure of all components:**

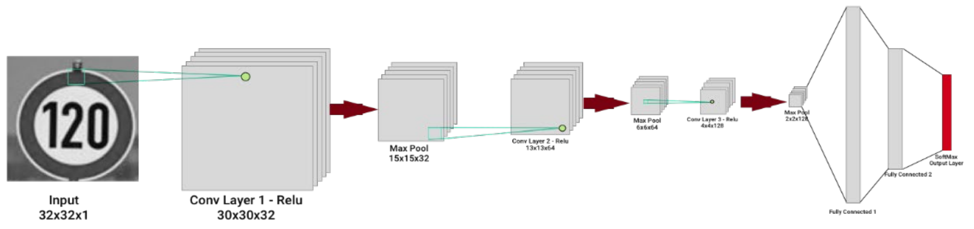


Figure 1: The above given figure is the image of EdLeNet 3*3 Architecture.

c) **Each step and model of your approach:**
The steps that I used for the approach include:

1) **<u>Step 1:</u>** First, the data set was divide into three sections for training, testing, and validation. The number of training images is 34799. The test images available are 12630. It also consists of 4410 validation images that are used to avoid overfitting. Then the images that are in 3 channels are converted to single grayscale. Now to make sure that the model treats all the images uniformly normalization is applied to the images.
2) **<u>Step 2:</u>** The model that I used is inspired by the LeNet Architecture which is presented by Yann Le Cun. I have added some tweaks to that architecture to provide more accurate results and the name of the network is EdLeNet. I have used a kernel size of 3*3 and started with a depth of 32. The network consists of 3 convolution layer and each of them is followed by a 2*2 max pooling. The network is trained using Adam optimizer. In order to improve the reliability of the model I have also used the dropout to improve the reliability and it also helps prevent overfitting. The model is trained based on the training data which consists of 34799 images. Now that images were shuffled so that the model is trained for all the classes and no classes get missed. It is also made sure that the images in the training dataset are not repeated in the testing dataset to check if the model is good for generalized images.
3) **<u>Step 3:</u>** Then the model is evaluated based on the confusion matrix and the testing accuracy of around 98% was observed. The graph of accuracy and loss was also printed so as to check if there is an anomaly. In the end, the images are printed along with their prediction and ground truth to check if the system is evaluating correctly or not.

The model consists of the following:

Step 1: The 1st Convolutional Layer
- Input = 32 x 32 x 1 and Output = 30 x 30 x 32
- A 3 x 3 Filter was used with output depth of 32 and a RELU Activation function to the output .
- Input for max pooling= 30 x 30 x 32 and Output = 15 x 15 x 32

Step 2: The 2nd Convolutional Layer
- Input =15 x 15 x 32 and Output = 13 x 13 x 64
- A 3 x 3 Filter with output depth of 64 and RELU Activation function to the output.
- Input for max pooling= 13 x 13 x 64 and Output = 6 x 6 x 64

Step 3: The 3rd Convolutional Layer

- Input =6 x 6 x 64 and Output = 4 x 4 x 128
- A 3 x 3 Filter with output depth of 128 and a RELU Activation function to the output.
- Input for max pooling= 4 x 4 x 128 and Output = 2 x 2 x 128
- After this a dropout is applied to avoid any overfitting

Step 4: Flattening The Network

- The network was flatten with Input = 2 x 2 x 128 and Output = 512

Step 5: 1st Fully Connected Layer

- A Fully Connected layer with Input = 512 and Output = 120 with RELU Activation function to the output.
- Then a dropout was again applied to get more reliable results.

Step 6: 2nd Fully Connected Layer

- A Fully Connected Layer with Input = 120 and Output = 84 with RELU Activation function to the output.

Step 7: 3rd Fully Connected Layer(The Output)

- A Fully Connected layer with Input = 84 and Output = 43 with Softmax axtivation function to the output.

d) Additional figures for illustrating certain steps or models:

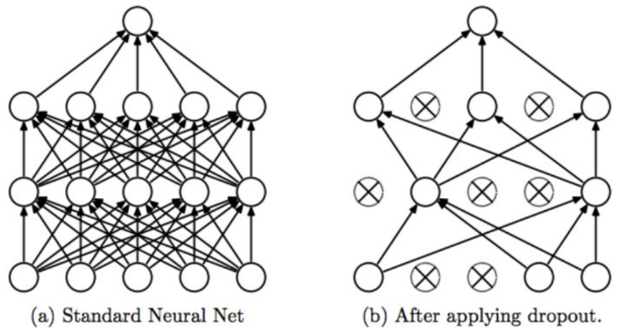


Figure 2: Dropuot

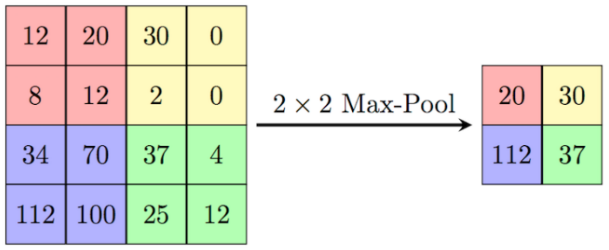
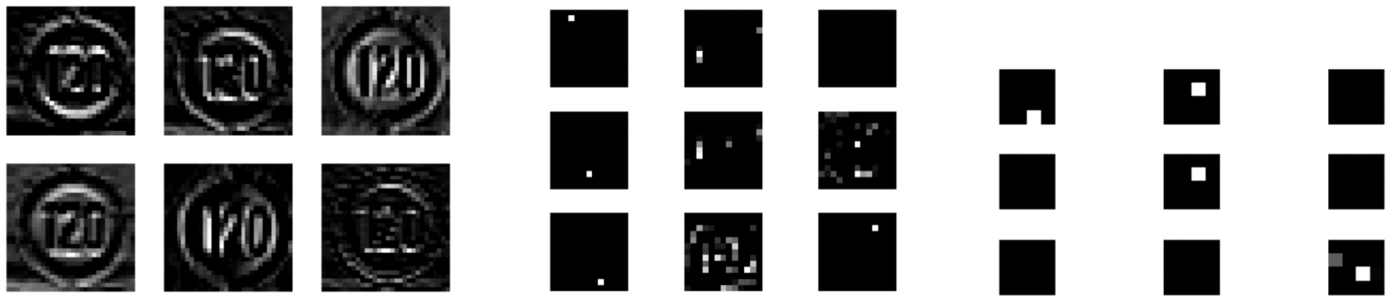


Figure 3: Max pooling

Figure 2 explains dropout which is a simple and powerful regularization technique for neural networks and deep learning models is a dropout. Deep learning neural networks are likely to quickly overfit a training dataset with few examples. Thus to prevent the model from overfitting we use the Dropout. Dropout can be implemented with all the layers. The dropout cannot be used on the output layer. Dropout is not used after training when making a prediction with the fit network.

Figure 3 tells us about the max-pooling which is a sample-based discretization process. The objective is to down-sample an input representation by reducing its dimensionality and allowing for assumptions to be made about features. As we can see in the image for each of the regions represented by the filter, we will take the max of that region and create a new, output matrix where each element is the max of a region in the original input.



a) Output from 1st Conv Layer b) Output from 2nd Conv Layer c) Output from 3rd Conv Layer

Figure 4: Output of Convolution layers

Now the above-given images are just a small illustration of how the images look after each iteration as they are been sampled down so as to obtain the best features from them which can later be used to compare it with the testing data.

3) Evaluation:

a) Implementation details:

The code is written in python and the software libraries used include keras, pandas, matplotlib, seaborn, NumPy and random and since the paper on LeNet architecture was published in 1998 there has been a lot of implementation of Traffic sign classification using LeNet Architecture but the open-source code and article I referred to was https://www.pyimagesearch.com/2019/11/04/traffic-sign-classification-with-keras-and-deep-learning/

The manually set input parameters include the number of key points for matching is 20 and the hyperparameters of deep learning are learning rate 0.001, batch size 512, number of training epochs 200 and the dropout keep percentage for the convolution layer and the fully connected layer was 0.5 as I was using the condition that the dropout value of the convolution layer should be greater than or equal to the fully connected layer.

b) Dataset:

For this project I will be using GTSRB - German Traffic Sign Recognition Benchmark on Kaggle. It consists of 43 different classes of images of signs on the road and more than 50,000 images in total the labels present are related to the training images. The number of training images is 34799. The test images available are 12630. It also consists of 4410 validation images that are used to avoid overfitting. The ground truth for each image is the label of class the different traffic sign belong to such as traffic signs along the road, speed limit signs, yield signs, merge signs and others.

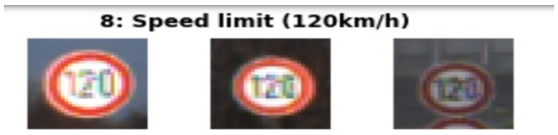


Figure 5: Snipet from the dataset

Figure 5 is a part of the dataset as you can see that the images can be barely seen in the images and if more contrast is added to them it will become difficult to identify them so to improve the reliability I used the dropout in the fully connected layers. This helped me achieve higher testing accuracy which makes my model better than the older version.

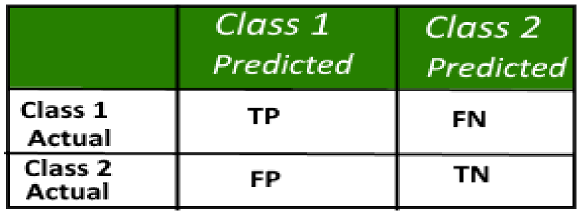
c) The evaluation metrics:

The evaluation matrix that I am using is the confusion matrix also known as error matrix. The main use of the confusion matrix is to describe the performance of a classification model. It allows us to visualize the performance of a model. A confusion matrix is a summary of prediction results on a classification problem. It gives us insight into not only the errors being made by a classifier but also the types of errors that are being made. If the diagonal of the matrix has more values present than it means that the model can predict a lot of positive values. Let us consider the below given example where class 1 is positive and class 2 is negative. We will be able to determine the classification accuracy using the formula:

Precision: Precision indicates the probability of the class that has been predicted matches with the ground truth of the image. Higher value of precision indicates better detection by the model. It is ratio of number of true positives to sum of true and false positives.

Recall: Recall indicates the ratio of true positives to sum of true positives and false negatives. It is extent with which the ground truth guessed correctly .

Accuracy= TP + TN /(TP + TN + FP + FN) **Recall**= TP /(TP + FN) **Precision**= TP /(TP + FP)



	Class 1 Predicted	Class 2 Predicted
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

d) Simpler versions of the approach(Baseline Approach):

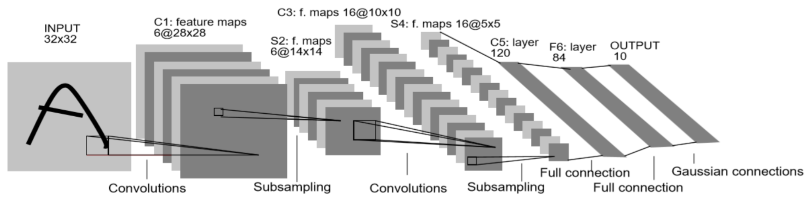


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

The LeNet Architecture by Yann Le Cun is the simpler version of the approach that I have selected. The name of my approach is EdLeNet Architecture. Some of the differences consist of a different number of convolution layers, different dimensions of fully connected layer, the addition of the dropout, max-pooling instead of average pooling, the depth sizes are different and we are doubling the depth each layer, different kernel size as well.An ablation study was conducted by removing certain of the model and see how the performance is affected by the full model(EdLeNet) vs the reduced model(LeNet)

1) The first feature removed was the third convolution layer after that it was easily observed that the difference between the training accuracy and validation accuracy has changed a lot from a mere 5% it grew to around 13% and also the test accuracy reduced from 98% to 93%.
2) The first feature removed was the third convolution and then after that the dropout layer was also removed which further affected the accuracy of the model eventhough the reduced model is still strong without the additional features the accuracy and reliability is enhanced by these features.

e) Table with the quantitative evaluation on the dataset:

The confusion matrix is used to evaluate the dataset and the accuracy of the classifier was reported to be 97.62%. The other way to look at it is that the diagonal of the confusion is darker than the other areas which mean that the model is able to correctly identify the class of the images based on the ground truth which consisted of the label of classes the images belong to. The precision and recall for all the classes was caluculated based on the confusion matrix and the average precesion was around 96% and the average recall was around 90% for my approach. Eventhough the recall was high which means that the class of some of the images was detected incorrectly but still we are able to reach an accuracy of around 98%

Parameters	**Reduced Model(LeNet)**	**Full Model(EdLeNet)**
Test Accuracy	93.2%	97.62%
Training Accuracy	96.76%	99.34%
Training Loss	0.013	0.0020
Validation Accuracy	88.48%	95.94%

Table 1: Comparision table between two models(LeNet and EdLeNet)

Even though both the models are able to achieve a good amount of accuracy but from the above table we can easily conclude that our model is better than the original model as we are able to predict the test data results more accurately and the accuracy of the validation data has also increased a lot and we are able to reduce the training loss as well.

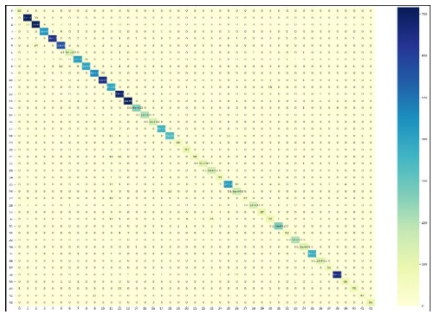


Fig 8: Confusion Matrix

The most crucial element for my approach I feel is the dropout layer as the number of hidden layers is still the same as the original model had 2 layers and my model has 3 layers I feel that the dropout has helped a lot in improving the accuracy and reliability of the model. Also, the use of max pooling has helped in extracting the most extreme features which are also useful.

f) Training runtime and test runtime per iteration and computer hardware:

I have used both the google colab server as well as my laptop with Intel i7 processor to run the model.

Hardware	Training(34799 images)	Classifying 12360 test images
Google colab	30 mins	2s
Intel core i7	32 mins	3s

Table 2: Hardware used

The main reason that I used both the model is to check if on both the hardware the model can learn at a good rate and also to see if there is any large difference in accuracy of the model but the difference is minimal.

g) Qualitative evaluation in figures:

To show the qualitative evaluation of the models I am using models with different contrasts so that I can assess if the model is accurate under different lighting conditions. The below-given images describe the qualitative evaluation of the images under different conditions. The Test accuracy that I was able to achieve 97.30% for images under different conditions. For this, I made use of data augmentation to get images under different lighting settings.

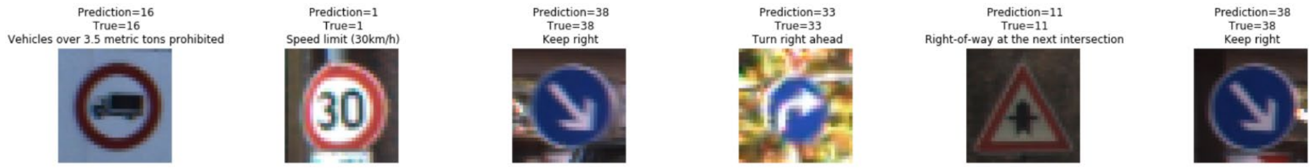
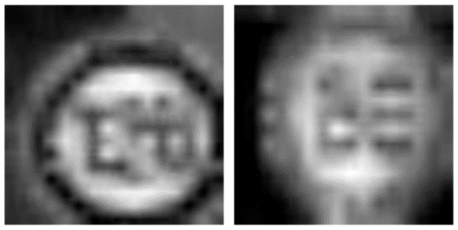


Fig 9 : Output of the model

However the accuracy doesn't receive 100% and some of the signs that were recognized incorrectly are given below:



I feel that even though I wasn't able to achieve better accuracy, even if the accuracy wasn't above 99% the identification of such images would be difficult. The main reason for the error is the low-resolution and also because the images are pixelated which is making it difficult to identify them. Even though I was able to overcome a lot of challenges I feel the low-resolution was difficult to completely overcome.

4) Detailed description of labor distribution across the team members

Since I was working on the project alone the work was done by me only. Firstly I changed the images from RGB to gray and then normalized the image. This helped the model treating images uniformly. The model that I used was EdLeNet which is an advanced version of the original LeNet with a few tweaks like different kernel sizes, different numbers of convolution layers, etc. Then the model was trained and tested using the confusion matrix. And all the abovementioned steps were executed only by me.