ROAD SIGN CLASSIFICATION

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

In

Computer Science and Engineering

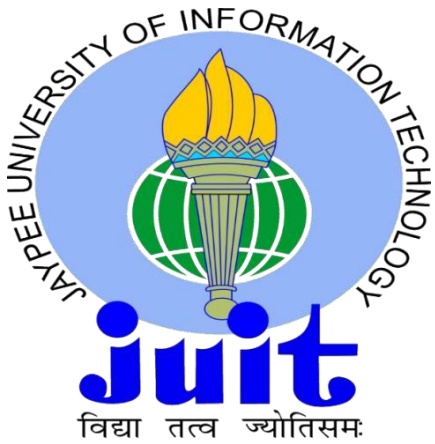
By

TARINI THAKUR (171378)

Under the supervision of

Dr. Pradeep Kumar Gupta

to



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled "Road Sign Classifier" in the partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted to the department of Computer Science and Information Technology, Jaypee University of Information Technology Waknaghat is a record of my own work and carried out over a period from January 2021 to May 2021 under the supervision of **Dr. Pradeep Kumar Gupta** (Associate Professor, Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Tarini Thakur (171378)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Pradeep Kumar Gupta
Associate Professor
Computer Science & Engineering and Information Technology
Dated:

ACKNOWLEDGEMENT

We would like to express our deep sense of gratitude and indebtedness to **Dr. Pradeep Kumar Gupta,** Department of Computer Science & Engineering and Information Technology, our supervisor for this work, for his guidance, support, motivation and encouragement throughout the completion of this project. His eagerness to listen to our problems, his educative comments, his advices for the timely and successful completion of this work has been exemplary.

We are grateful to **Prof. Dr. Samir Dev Gupta**, Head of Dept. of CSE and IT for his excellent support during our work. We would also like to thank all the professors and other supporting members of the department of Computer Science & Engineering and Information Technology for their generous help in various ways for the completion of this work.

We are also thankful to **Mr. Ravi Raina and Mr. Rohit Sharma,** our Lab Assistant, for his support and guidance throughout the process.

TABLE OF CONTENT

ABSTRACT

To maintain rule and order on roads and to make sure the security of citizens on road traffic signs are used to help the drivers know the type of the road, in what speed he should travel what all ways he should adhere to while being on road to prevent any road accidents and to make sure that the road journey is safe for everybody. But the road signs used can be of different dimensions,color,shape and size and may be mistakenly interpreted in a different manner than what it intends to especially during night time .Our task is to train our system in such a way that it is capable of identify and classify the road signs correctly.

For the purpose we will be using a deep convolution neural network that will help our system to classify the images of traffic sign correctly .For learning purposes I have developed end to end model and also deployed it using flask.

1. Chapter 1. INTRODUCTION

1.1 Introduction

Road Sign Classification is a primary part of any self-governing driving structure. The movement of vehicles in the near potential depends on such an arrangement for harmless routing among the traffic that will consist of humans and other self-driving vehicles.

To maintain rule and order on roads and to make sure the security of citizens on road traffic signs are used to help the drivers know the type of the road, in what speed he should travel what all ways he should adhere to while being on road to prevent any road accidents and to make sure that the road journey is safe for everybody. But the road signs used can be of different dimensions,color,shape and size and may be mistakenly interpreted in a different manner than what it intends to especially during night time .Our task is to train our system in such a way that it is capable of identify and classify the road signs correctly.

The elementary intention of this particular idea is to devise and build up a strong and computationally lightweight model, which could precisely classify the road signs used to prevent traffic. Most of the early works done in this field includes the use of conservative method of computer vision and classification. In our project we have tried to use deep convolutional neural network in order to achieve a accuracy in classifying images and also tried to deploy it so that it could be easily deployed on the new generations car hardware.

1.2 PROBLEM STATEMENT

The project aims to devise and build up a strong and computationally lightweight model, which could precisely classify the road signs used to prevent traffic.We have also done the Exploratory data analysis initially to umderstand the data. Most of the early works done in this field includes the use of conservative method of computer vision and classification. In our project we have tried to use deep convolutional neural network in order to achieve a accuracy in classifying images and also tried to deploy it so that it could be easily deployed on the new generations car hardware.

It has very chief responsibility to play in self-driving cars, which is definitely the upcoming technology in the field of automobile industry.

1.3 Tech-Stack:

1. Python 3.7
2. Jupyter Notebook
3. Libraries:
    - Keras
    - Tensorflow
    - Sklearn
    - Numpy
    - Pandas
    - Matplotlib
    - PIL
    - OpenCv
    - Flask
    - os

1.4 DATASET:

We have downloaded the dataset from kaggle: German Traffic Sign recognition Benchmark. It consists of about 50,000 images and there are forty-three classes. The dataset downloaded consists of Train and Test folders which contain the pictures for the purpose of training testing, resp. The train foder contains 43 subfoders contaning images beloging in a particular class. We also have in the dataset the meta folder that contains forty-three images that ranges from 0-42 and basically consists of images from the class corresponding to it.

The dataset also comproises of 3 csv files that basically comprises of the various proportions, path of each icon in their individual folders and ClassIDs which symbolize its corresponding class.

1.5 METHODOLOGY

This Project follows this framework:

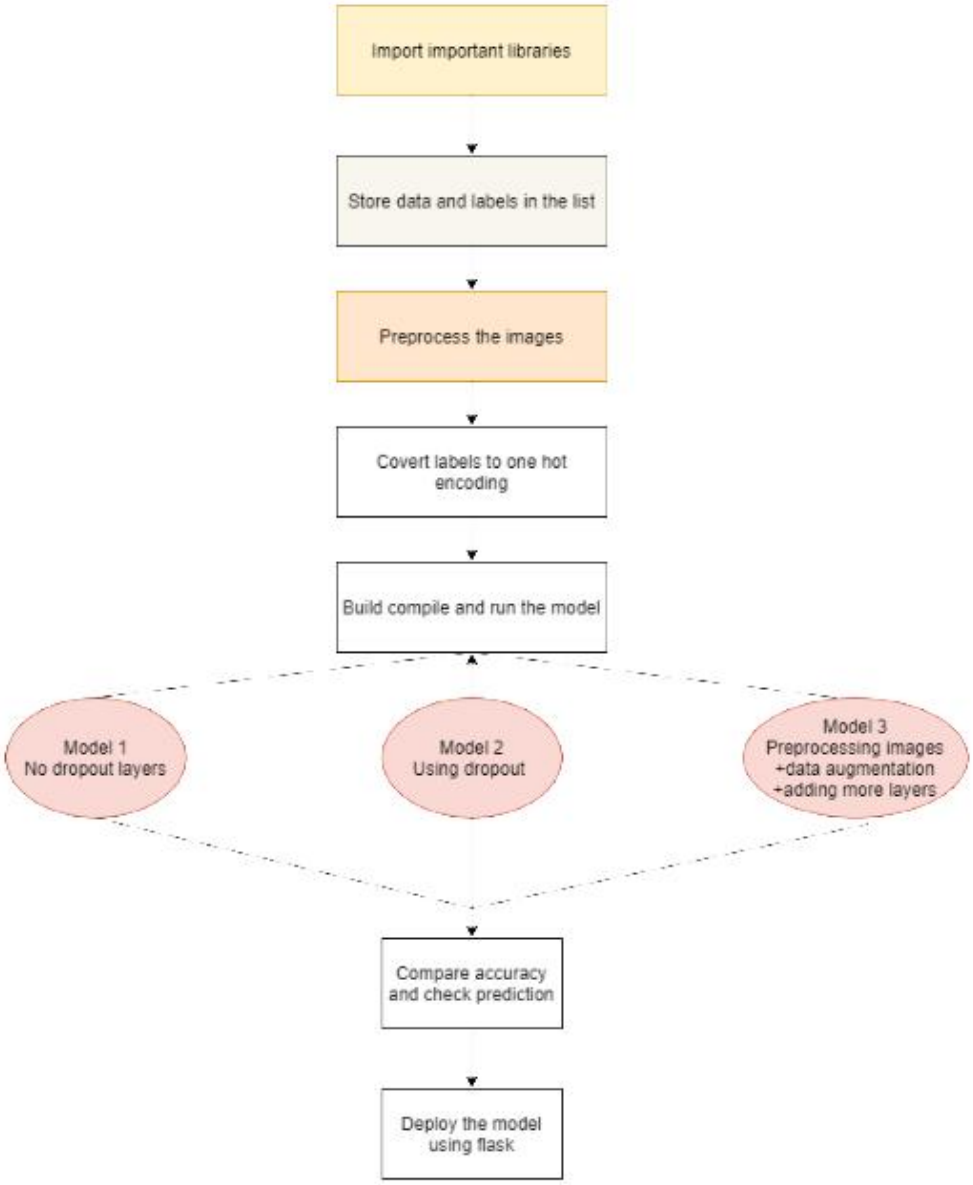


Figure1. Framework used in the project

2. Chapter 2 LITERATURE SURVEY

Road Sign Classification is a primary part of any self-governing driving structure. Passage symbols present the drivers with very priceless infomation regarding the road, to make sure that the task of driving becomes more safe and trouble-free. The progress of vehicles in the near potential depends on such an arrangement for harmless routing among the traffic that will consist of humans and other self-driving vehicles.

In the past the need for the machines to detect objects and be able to classify them has been observed and a lot of work has been carried out to support the same. Arturo de la Escalera [1] was inspired to study the field of road traffic signs. The procedure followed by him for detection and classification involves two segments .One of the segments was designed for the purpose of recognition, and in this he used color thesholding to fragment the picture and outline for the purpose of rightfully detecting the icons. The other segment was for the purpose of categorization for which he used the neual netwoks. The limitations of this paper were that the researchers were not able to sense and correctly categorize the yield sign that consists of the upturned triangle and the stop sign of hexagon outline.

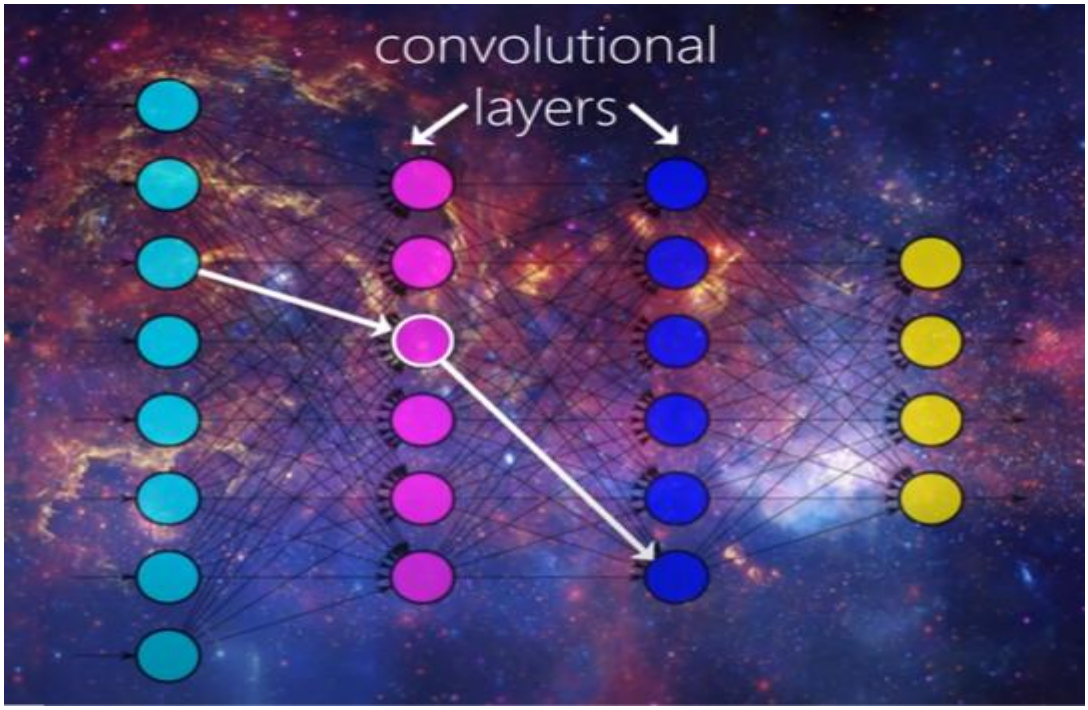
In another study Sebastian Houben [2] tried to divide sign detection from that of classification, and have measured the presentation on pertinent category of symbols to be able to find the standardize dedicated solution. The procedure followed by them represents the methods which are widely popular such as the Viola-Jones detectr which is based on the featurs and a linear classifier which relies on Histogram of Oriented Gradients descriptors. They have used binary loss function in the algos used for the purpose of object detection. They have used three methods one is the viola joness detector ,the other is hog descriptors and other model based approaches. They have also considered a special case of detecting and classification of danger symbols for which they performed projective adjustment to the ROIs and re-classify them with Histogram of Oriented Gradients and SVM.

3. CHAPTER 3. SYSTEM DEVELOPMENT

ALGORITHM USED: Convolution Neural Network

The algorithm that is being used in our project is Convolution Neural Network which is also popularly known as CC/Convnet. CNN is an artificial intelligence neural network that is so far being widely used for analysis of images. Although image analysis is the most widespread use of CNN they can also be used for other data analysis or classification problems as well.

CNN is an Artificial neural Network that has same type of specialization for being able to pick out or detect patterns and make sense out of them. This pattern detection is what makes CNN so useful for image analysis.



CNN has hidden layers/convolution layer. These layers are what make it CNN. Convolution Neural Network can have and usually they do have other non convolutional layers as well, but basic feature of it is convolutional layers.

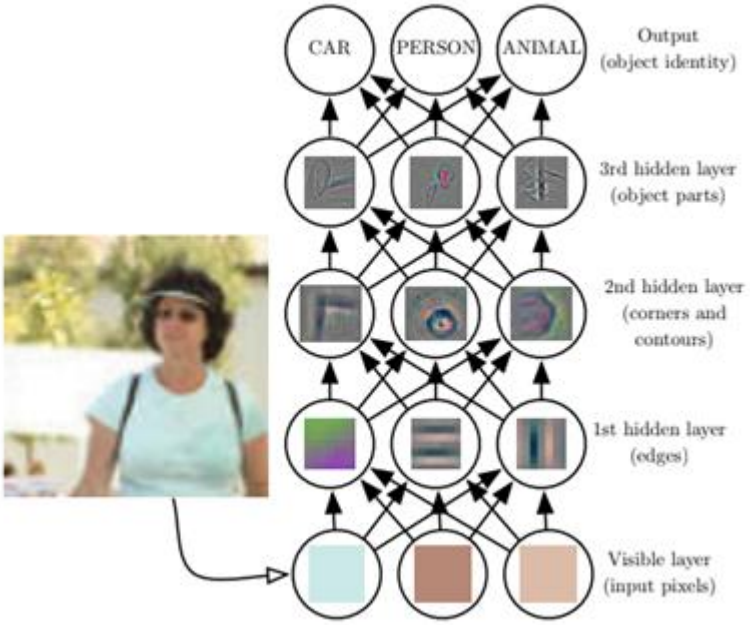
WHAT DO CONVOLUTION LAYERS DO?

Convolution layers, just like any other layer of neural network initially receives

input, and then transforms the input it receives in some way and then it outputs the transformed input to next layer.

With each convolution layer we need to specify the number of filters the layer should have. These filters are responsible for detecting patterns in images.

Deeper layers are able to detect more sophisticated images like complete dog, cat, etc.



Let us understand the working of hidden layers

With the help of deep learning we can train the neural network with a set of examples and we can find common features between images. We can extract high level features like ear, nose and other object parts.

Function of hidden layers: By using these hidden layers we can extract minute details from the image to get a proper image by identifying all the edges, contours and corners and the object parts.

Visible layer: In this input layer we will feed the layer with pixel values of the individual pixels in each neuron.

1st Hidden Layer: In this layer we detect low-level features like edges and if any curved lines are present in the picture are detected.

2nd Hidden Layer: In this layer we combine the edges that we detected in the previous layer and with the help of that we find features like corners and contours and the information is fed to the next layer.

3rd Hidden Layer: Now by combining the corners and contours etc different objects like ears, eyes, and nose are detected

Output: Now by combining different parts the object is identified if it is a cat, dog, human etc.

From this we can understand that with the help of deep learning algorithms learn the features by itself when we train it with huge amounts of data without any human intervention.

4. EXPLORATORY DATA ANALYSYIS

1. Let us see the shape of our dataset

```
print("No of images in train data : {0}".format(df_train.shape[0]))
print("No of images in test data : {0}".format(df_test.shape[0]))
```

```
No of images in train data : 39209
No of images in test data : 12630
```

Fig 2. Shape of dataset

2. In next step we will see if the train data is balanced or not

```
fig, ax = plt.subplots(1, 1)
fig.set_size_inches((25,10))
sns.countplot(data = df_train, x = 'ClassId',color='pink' )
plt.show()
```

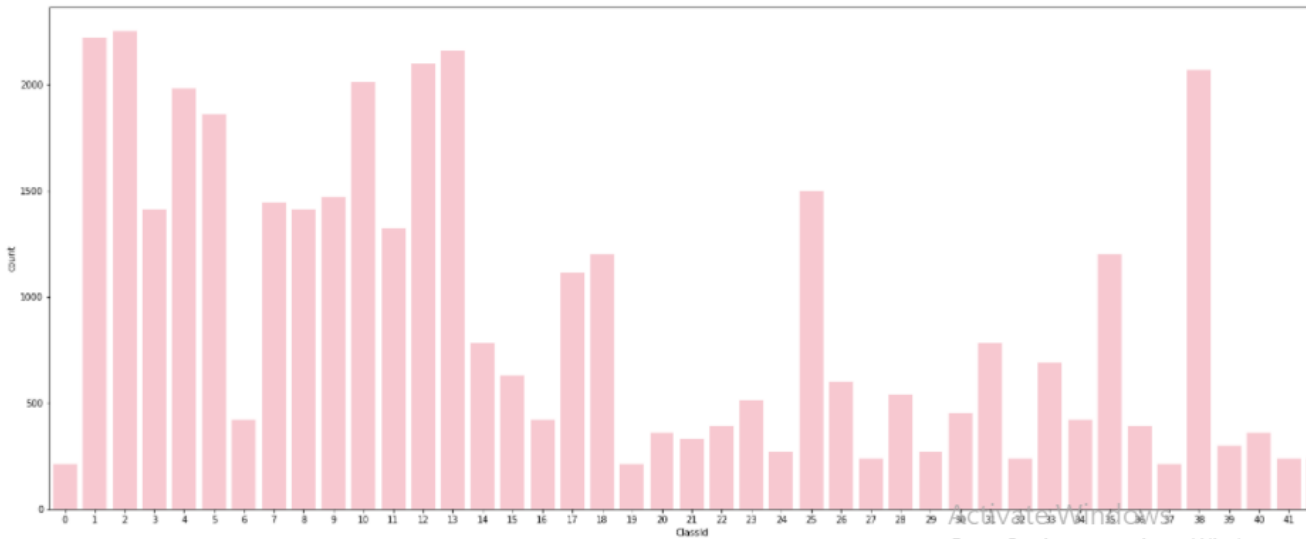


Fig 3. Graph for class imbalance

From Fig 3.we have deduced that the 42 classes we are dealing with are not for sure concentrated in a regular way. Also we can observe from the graph that the irregularity is not so much not so stern between the dissimilar classes.

3. In next step we will see if the test data is balanced or not

```
fig, ax = plt.subplots(1, 1)
fig.set_size_inches((25,10))
sns.countplot(data = df_test, x = 'ClassId',color='green' )
plt.show()
```

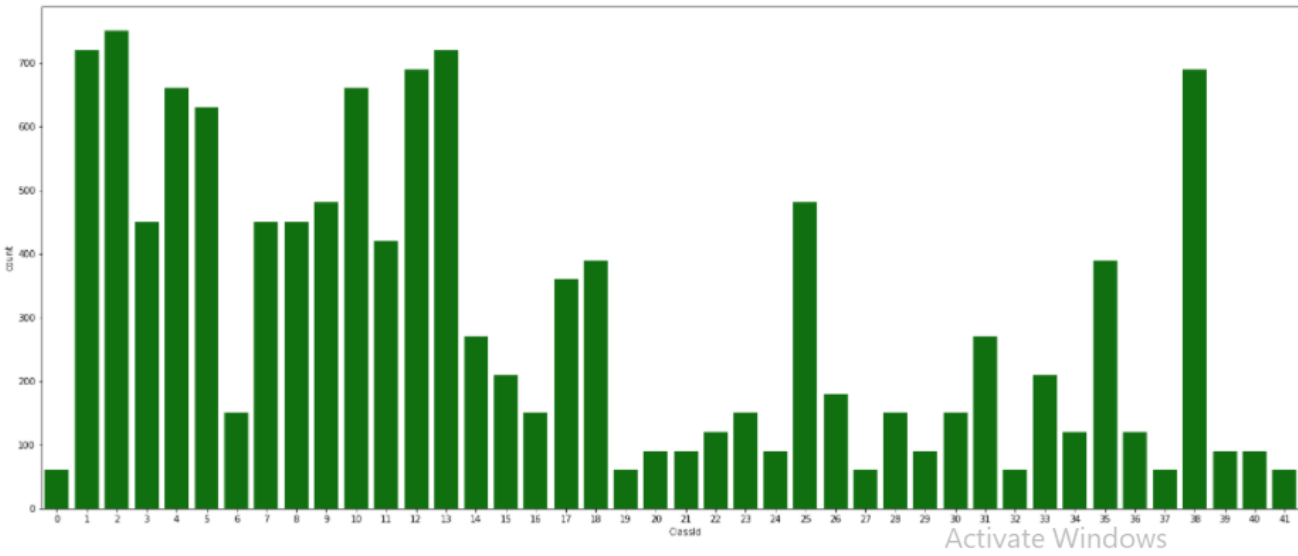
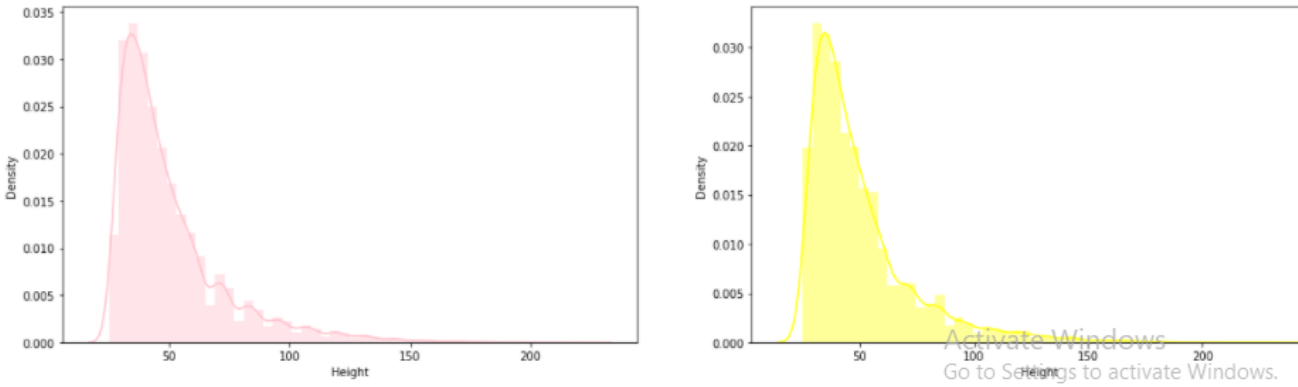


Fig.4 Balance in test data

From Fig.4 we can conclude that the train data and the test data are quite similar. There is not too much of difference between test and train data.

4. Processing images with different dimensions

```
fig, ax = plt.subplots(2, 2)
fig.set_size_inches((20,12))
sns.distplot(df_train['Height'], ax = ax[0][0],color='pink')
fig.set_size_inches((20,12))
sns.distplot(df_test['Height'], ax = ax[0][1],color='yellow')
fig.set_size_inches((20,12))
sns.distplot(df_train['Width'], ax = ax[1][0],color='orange')
fig.set_size_inches((20,12))
sns.distplot(df_test['Width'], ax = ax[1][1])
plt.show()
```



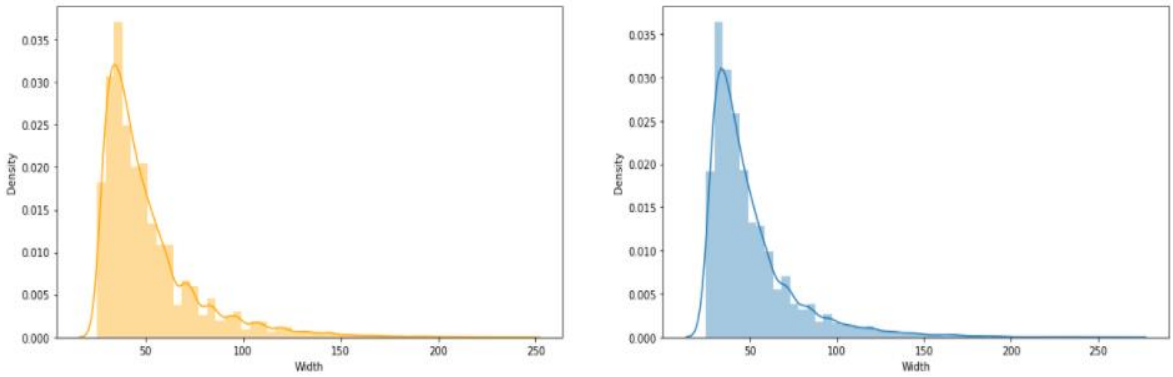


Fig.5 Distplot for Height and Width of images in train data

From above figure five we can observe that dataset used for training as well for testing has depicted similar pattern in their elevation and breadth. Also we can see since dimensions are different so in order to get efficient results ad in order to train our model efficiently we will have to make sure that the measurement of all images is same. In pre processing we have to careful and make sure that we do not incur any loss in data. In case we encounter small images we can bring it to same size using padding.

```
image = cv2.copyMakeBorder(image, top, bottom, left, right, cv2.
print(image.shape)
image = Image.fromarray(image)
plt.imshow(image)
plt.show()
```

```
(128, 128, 3)
```

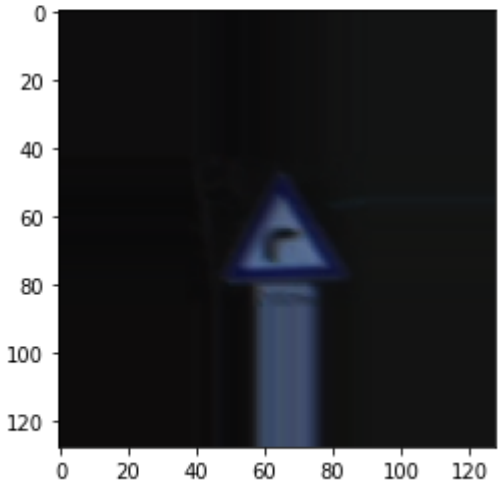


Fig 6 Dimension (128*128)

We are basically bringing all images to same dimensions.

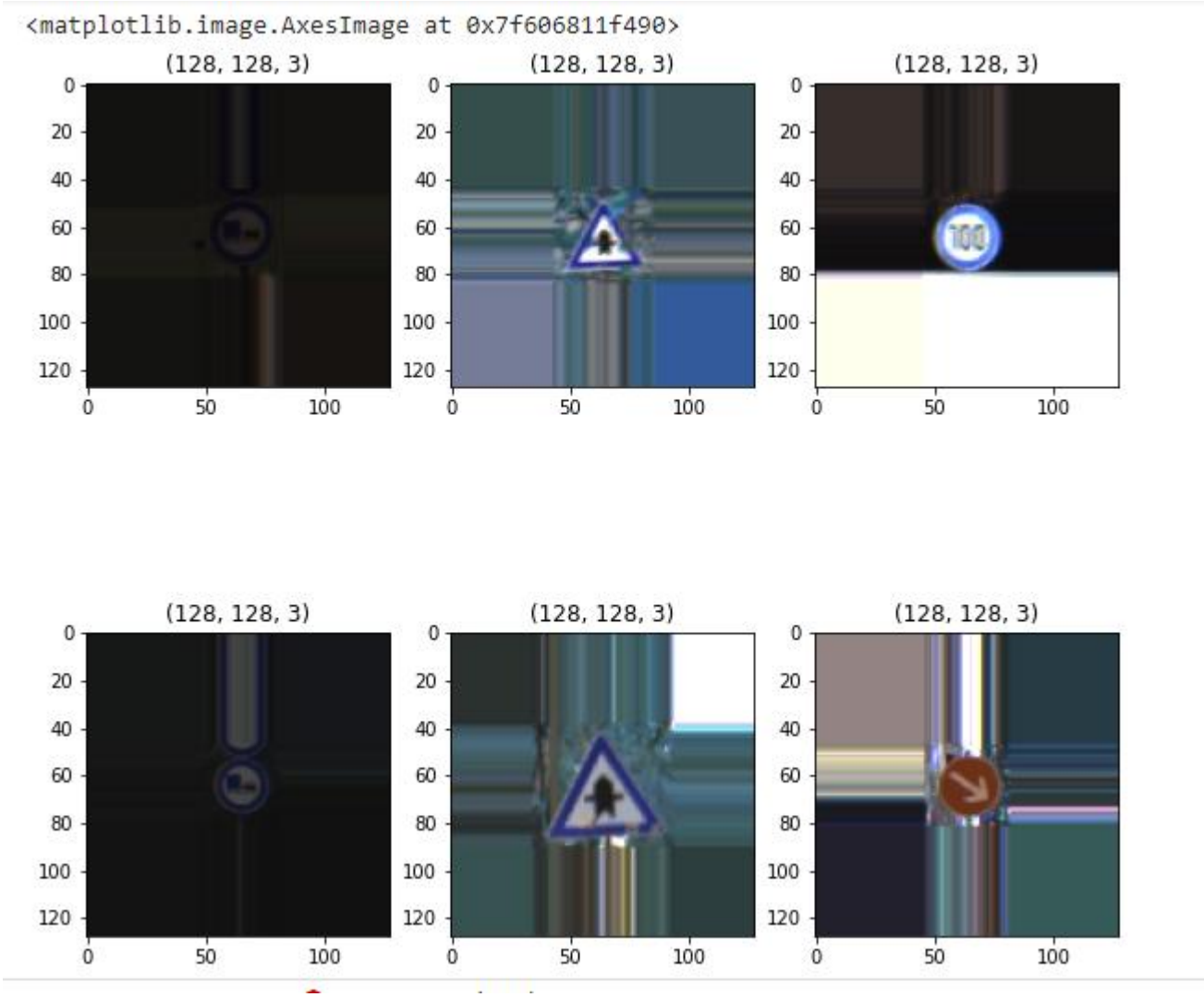


Fig.7 Preprocessed Images

Images seem not clear as small images have enlarged because of padding, which we will correct with the aid of data augmentation by brightening the dark descriptions and build them visibly more understandable. Right now our aim of getting all images same dimensions has been achieve.

Fig 8 Brightness of Images

We can see that some pictures in the dataset are very murky while many are illustrious; using data augmentation we will tackle this predicament.

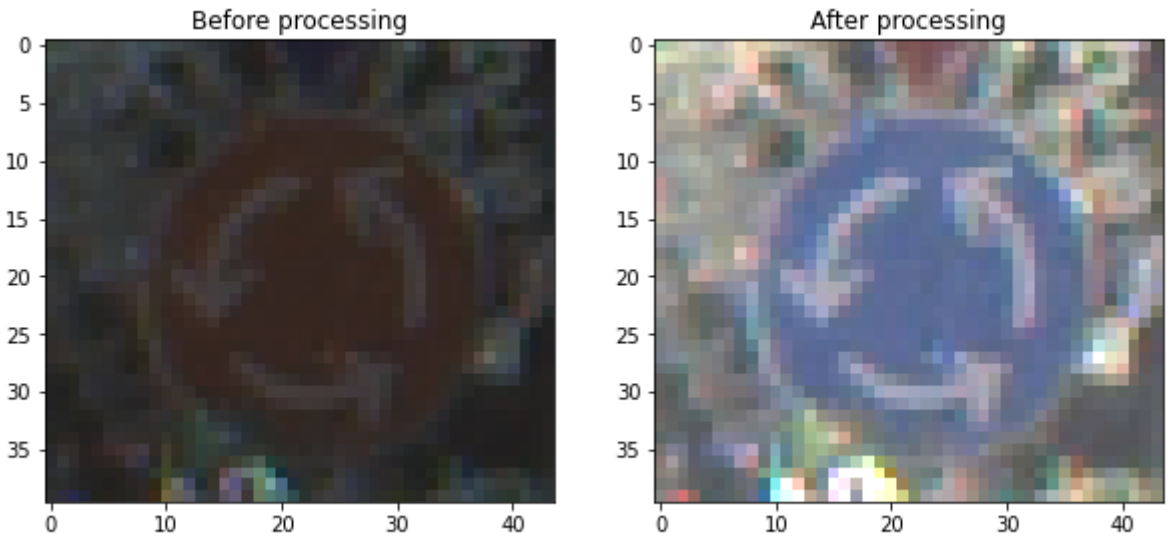


Fig 9. Image after processing

Using the preprocessing technique of enhancing the image as can be seen in the result produced above , I will be able to obtain a enhanced version of the picture, and using PIL we can also control the contrast and the brightnes oof the image.

For the purpose we have used Image Data Generator from keras.preprocessing.image to obtain various forms of every sign that we give as input. As a result we can obtain larger dataset and also various versions of every picture we are dealing with. This enhance our model and will also help it to understand and perform better and give us significant results.

Deploying Model Using Flask

Flask is basically a trivial framework for web which is basically writen in python. It is python Framewok. It is often introduces as a micro web framework because it is such a framework which is not dependent and does not need the use of any specific libraries or assets. For lightweit process it is best tool to use. Flask does not require any database abstraction cover for justification or for any other mechanism in which already existing third_party libraries supply frequent fuctions.

There are certain basic elemenys of flask that makes it special.Flask provides for a integated purpose of unit testing. It also has a feature of RESTful request disptching. Ninja2 template engine is used in flask web framework and is solely has a base of Werkze toolkit. It has a feature which is responsible for protecting cookies of the client of the client while hosting a session. It has got a lot of credentials. One of the best features of flask is that it also provides for compatibility with the Google app engine. Also with this framework we can develop APIs which are adequately formed and logical and the most important element that makes it one of the most widely used frameworks is because of its nature and ease in deploying any model using it. It definitely extend maintain for apis. One of the limitations with flask is it lacks providing us platform to create dynamic html page.

This particular framework is fitting for lone purpose an as mentioned earlier the most usefull elements of flask is its lightweightness and that it is a open source and most importantly does not require a lot of coding to be done for developing and creating a web app.

Other advantages of using this particular framework are that it is highly compatible in nature with the technologies in trend and is one of the easiest frameworks to be used for easy cases. Also does not require lot of coding to be done and is highly scalable for developing simple applications as well to build easily and quickly a prototype. One of the best thing it provides for is that with it we can easily route the URL. It is simple to build up and preserve the built applications and also integrating database is a feasible task. It has a minute core and with no trouble extensible.

It is a very powerful platform in deploying the models and also if any help require in learning widely ranged data and resources are available on the internet for this purpose.

Now I will try to present the working of this framework. we need to have one index.html file, this index.html file basically is required and this acts as a front end web app so that any request that we give to the model which will be in the form of API which we will host through flask will basically interact with that and get the output from the particular API itself.

We have also attempted to do some styling also .Then next file that is required is the traffic-sign-app.py file; this file is the one that basically contains the written flask code. Also another important step required is to save our already trained model in order to prevent from running it again and again, saving the model will save us a lot of time. We have already saved our model in .h5 format.

We have also used render_template this basically will help us to redirect to the first home page that we actually created.

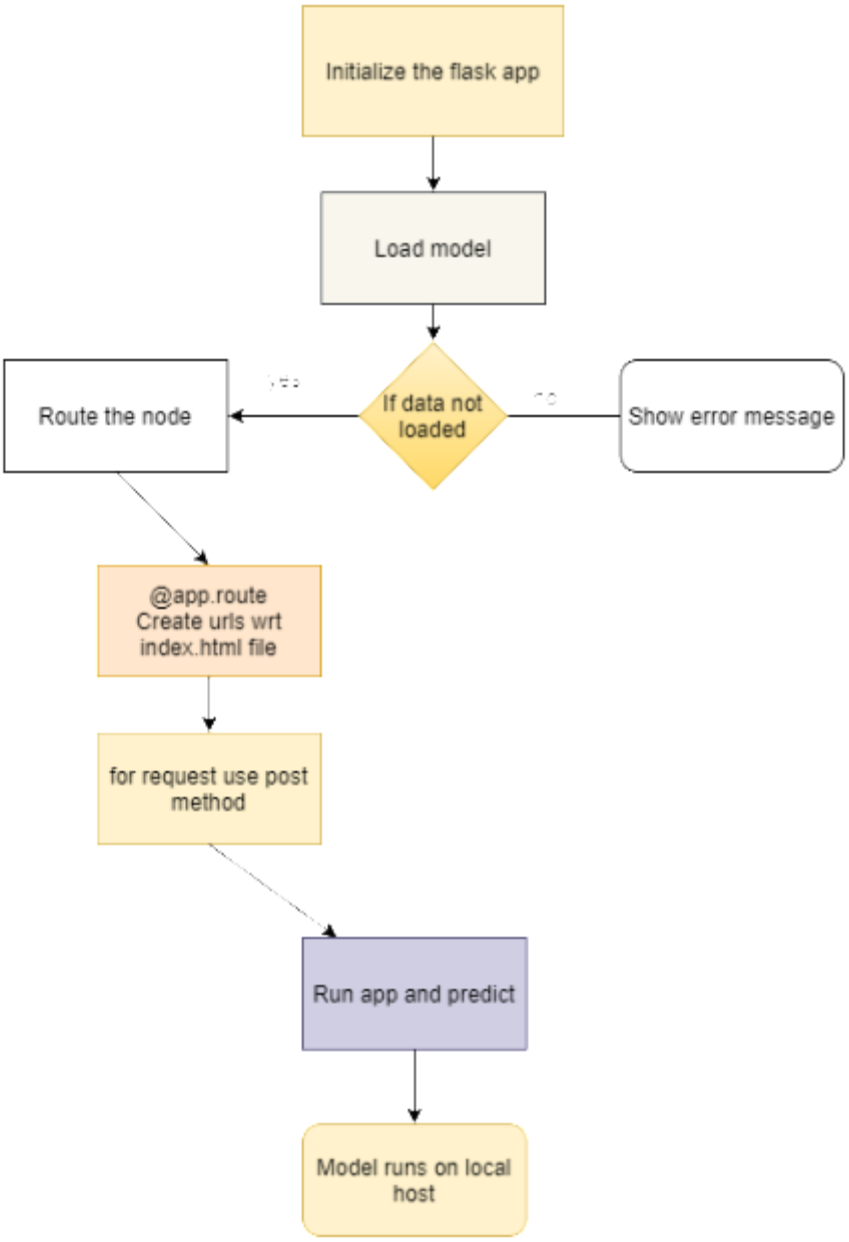


Fig 10 . Deploying using Flask

Applications

Road signs are basically for the purpose of caution and for creating a warning as well as to guide the drivers. Traffic signals are also helpful in regulating the flow of travel of the vehicles and the pedestrians or the motorcycles and also for bicycles and for all those who travel on the streets, highways or any other roadways.

Road signal classification and recogniion will be playing an important role in the advanced specialist systems like the vehicles in which we can provide for assistance generate alerts at a time when traffic signs are not that clearly visible and the self driving vehicles. It will be capable of assisting the drivers or even the self driving vehicles being developed in not only spotting but also identifying and then successfully classifying the road symbols efficiently.

5. PERFORMANCE ANALYSIS

We created three models and observed the development and the improvement in the accuracy obtained.

```
[INFO] Model Summary
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 32)        2432
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 24, 24, 32)        25632
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 12, 12, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 10, 10, 64)        18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 64)          36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 4, 4, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 256)               262400
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 43)                11051
=================================================================
Total params: 356,939
Trainable params: 356,939
Non-trainable params: 0
_________________________________________________________________
```

FIG 11. Model information

Multiple iterations on a simple model with little image preprocessing and no overfit mitigation strategies resulted in the model mentioned in the preceding section. Model 1 with no dropout layers, Model 2 with one dropout layer, and Model 3 with enhanced image preprocessing and augmentation as well as dropout layers are all evaluated in this portion.

MODEL 1

In model 1 we have not used any dropout layer and seven convolution layers .In the image processing all images were first brought into standard size of 30 pixels.The accuracy obtained in this model was descent which was about 90%.

Model 1 is a straightforward seven-layer model with no dropout layers. It had a preparation exactness of around 94%, however its testing precision was just 90% on the test range, demonstrating that it was overfitting.

Overfitting alludes to a model that models the preparation information excessively well. When a model knows the detail and clamour in the preparation details to the point that it negatively impacts the model's presentation on new data, this is known as overfitting. This means that the model gets and educates the commotion or arbitrary vacillations in the planning details as ideas. The issue is that these ideas don't matter to new information and contrarily sway the models capacity to sum up.

```
6 plt.show()
```

```
Predicted traffic sign is:  Road work
```

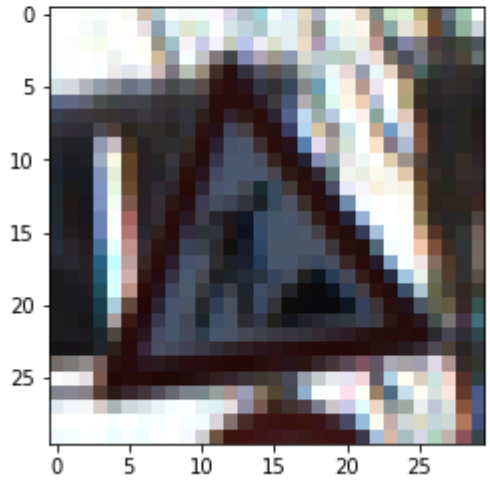
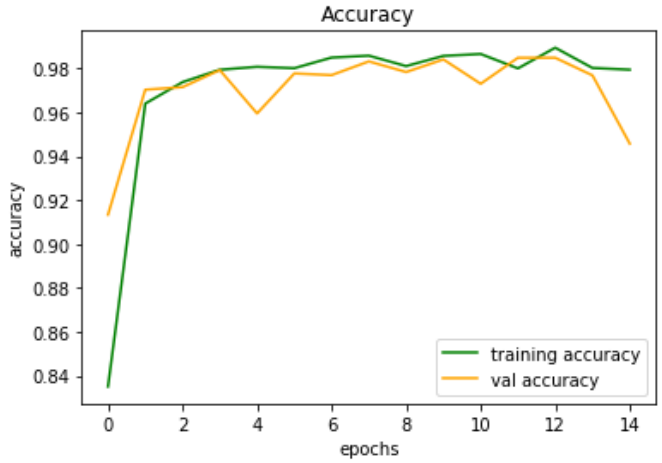
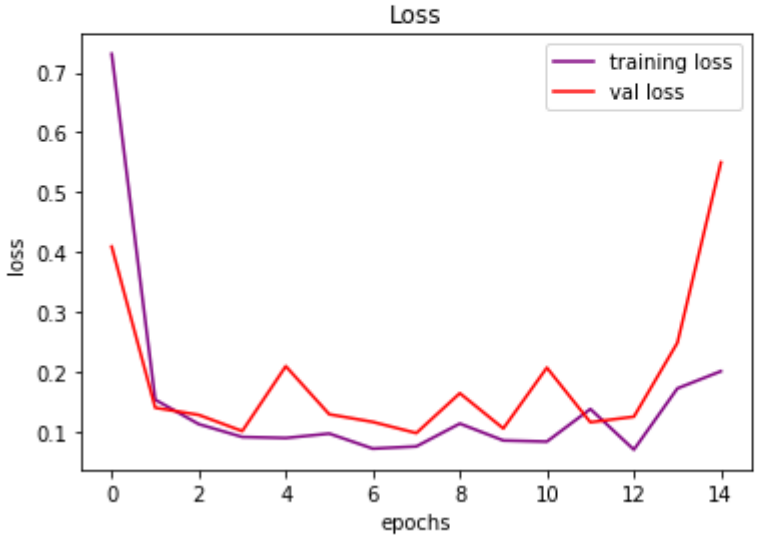


Fig 12. Plot for accuracy and loss of Model 1

In [98]:
```
# accuracy
plt.figure(0)
plt.plot(history.history['accuracy'], label='training accuracy',color='green')
plt.plot(history.history['val_accuracy'], label='val accuracy',color='orange')
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()
plt.show()
```




```
plt.plot(history.history['loss'], label='training loss',color='purple')
plt.plot(history.history['val_loss'], label='val loss',color='red')
plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```



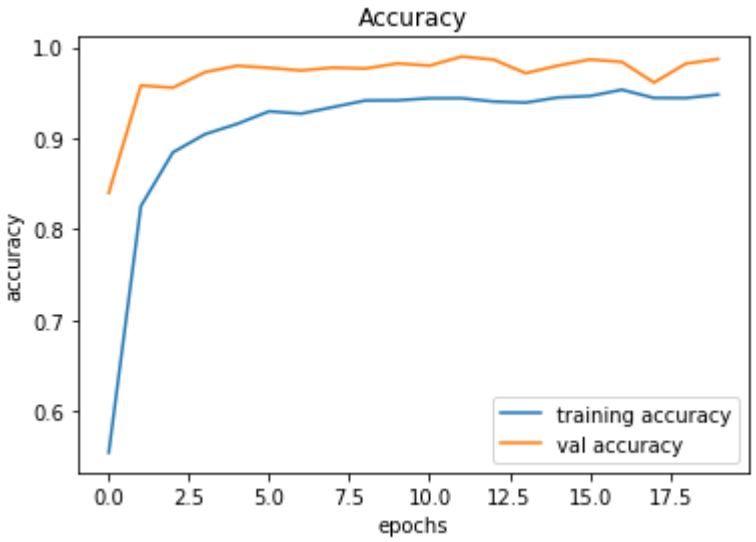
MODEL 2

I learned and checked Model 2 again, adding two more Conv2D layers as well as a dropout layer. On the test range, there was a notable improvement in precision, which was now about 94 percent.

The Dropout layer, which helps avoid overfitting, sets input units to 0 at random with a rate frequency at each phase during training time. When training is set to True, no values are lowered during inference, and the dropout layer is used. In a neural network, dropout is implemented per layer. Dropout can be applied to any or more of the network's hidden layers as well as the visible layers.

Another hyperparameter is added, which decides the likelihood of the layer's yields being exited, or, then again, the likelihood of the layer's yields being held. The understanding is a specialized detail that differs starting with one paper then onto the next.

For retaining the output of each node in a hidden layer, a probability of 0.5 is normal, and a value close to 1.0, such as 0.8, is common for retaining inputs from the visible layer.



In [19]:

```
# Loss
plt.plot(history.history['loss'], label='training loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```

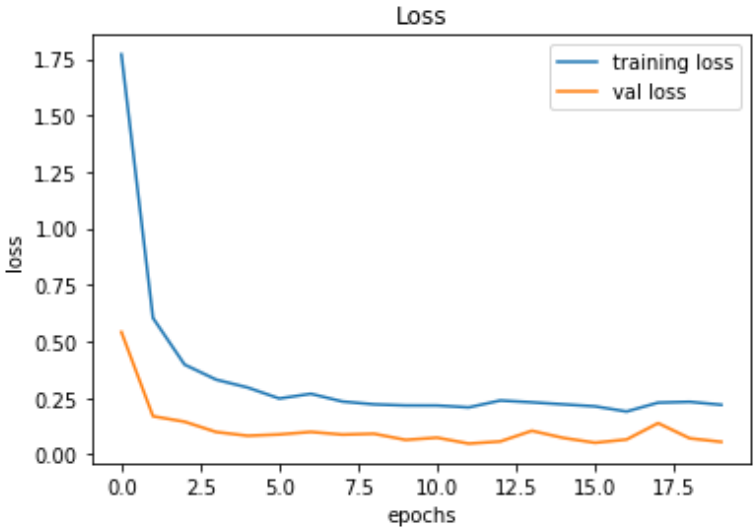


Fig 13. Accuracy and Loss Plot for Model 2

Model 3

At last, for Model 3, I utilized picture pre-handling methods including CLAHE change to upgrade difference and splendor, just as the picture information increase talked about in segment II. I likewise expanded the quantity of dropout layers in the model by two. On the test range, this model had the most elevated precision of 98%.

These outcomes were empowering as they affirmed that great picture preprocessing procedures can improve a model's precision and improve forecasts on pictures that were beforehand misclassified. Adding the dropout layers had likewise assisted with forestalling overfitting that was available in Model 1and 2.

On the validation range, the model is 99 percent accurate, and on the testing set, it is 98 percent accurate. On a 4-core CPU with no GPU acceleration, the model takes about 0.1s to process.

For image pre processing used CLAHE transformation. Since almost all of the images are different sizes, I resized them to 32x32 pixels before doing any image processing. Then I found the photos' poor lighting, contrast, and brightness. This is a recurring theme in the dataset. To solve this problem, I used the Scikit-Image library's Contrast Limited Adaptive Histogram Equalization transformation.

Next step involved was of data augmentation.

Without actually gathering new data, data augmentation allows clinicians to greatly expand the diversity of data available for training models. Cropping, padding, and horizontal flipping are popular data augmentation techniques used to train large neural networks.

Another factor to keep in mind is that most of the photos have a decent viewing angle, with the majority of them showing the sign from the front, which is useful for training but does not accurately represent real-life circumstances. As a result, a little image augmentation could make the model more resilient, as the augmentations would help the model mimic real-life scenarios. I utilized the ImageDataGenerator module from TensorFlow library to characterize the expansion.

ZCA WIGHTENING

ZCA Brightening is a picture preprocessing strategy that changes information into decorrelated includes by changing the covariance lattice to the personality network.

A brightening change or sphering change is a direct change that changes a vector of irregular factors with a known covariance framework into a bunch of new factors whose covariance is the personality grid, implying that they are uncorrelated and each have difference. The change is designated "brightening" on the grounds that it changes the info vector into a background noise.

Actual Changes on pivot zoom, shearing, and measurement – Changing these characteristics assisted with addressing certifiable situations where a sign may be harmed or turned, or the picture caught from a camera probably won't have a decent review point. No even or vertical flips are required on the grounds that, all things considered, signs won't be turned over.

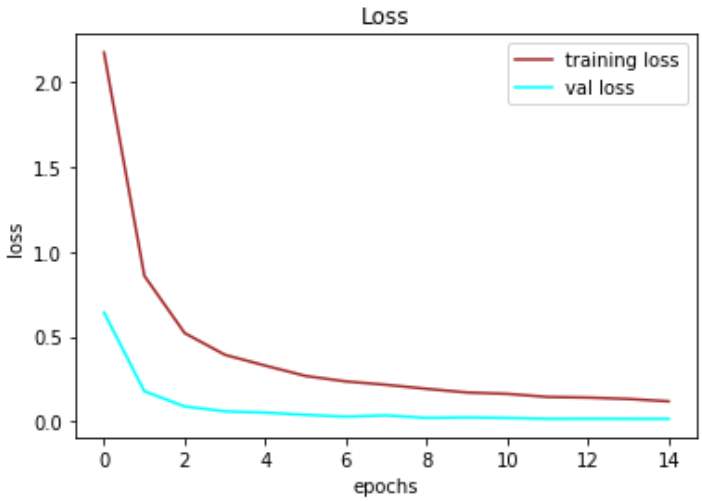
Shading Movements – I had a go at grayscaling as a preprocessing on the pictures, however it exacerbated the issue of low differentiation/splendor, so I went for a minor shading shift all things considered, so the model isn't overfitted to one tone or tint.

The human visual framework can recognize countless diverse shading shades and forces, yet just around 100 shades of dim. Subsequently, in a picture, a lot of additional data might be contained in the shading, and this additional data would then be able to be utilized to improve on picture investigation, for example object recognizable proof and extraction dependent on shading.

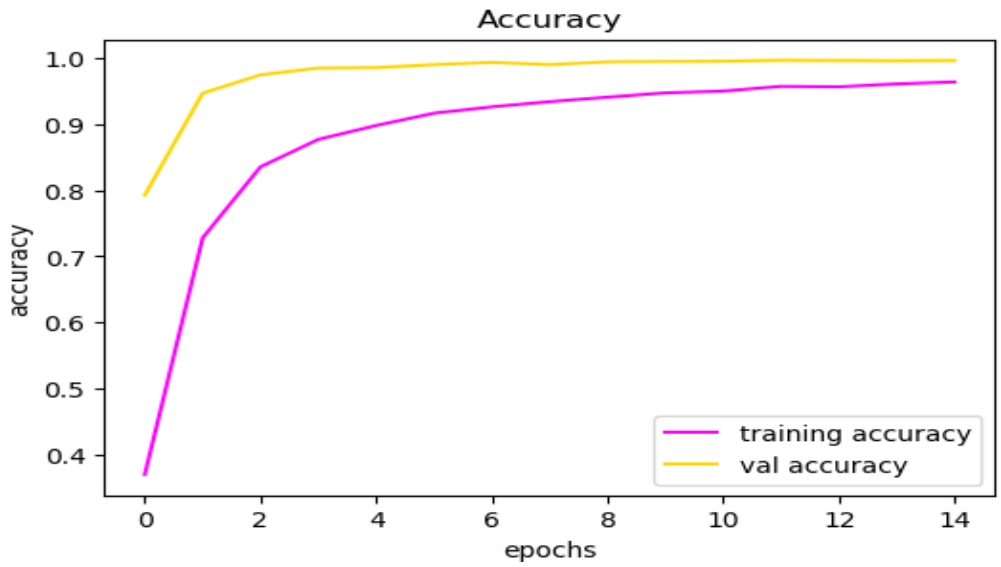
Three autonomous amounts are utilized to portray a specific tone. The tint is dictated by the prevailing frequency. Noticeable tones happen between about 400nm (violet) and 700nm (red) on the electromagnetic range.

FIG 14. Plot of accuracy and loss for model 3

In [27]:
```
plt.figure(1)
plt.plot(history.history['loss'], label='training loss',color='brown')
plt.plot(history.history['val_loss'], label='val loss',color='cyan')
plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```




```
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()
plt.show()
```



RESULTS OBTAINED

Model	ACCURACY(%)
Model1	90.31
Model2	94.52
Model 3	97.94

Fig 14 Results

Creating A user interface and deployment of model using flask

The model that I deployed using flask is presently running on local host . At that point next record that is required is the traffic-sign-app.py document; this document is the one that fundamentally contains the composed jar code. Likewise another significant advance required is to save our all around prepared model to keep from running it over and over, saving the model will save us a ton of time. We have effectively saved our model in .h5 design.

We have likewise utilized render_template this fundamentally will assist us with diverting to the main landing page that we really made.

Results Obtained

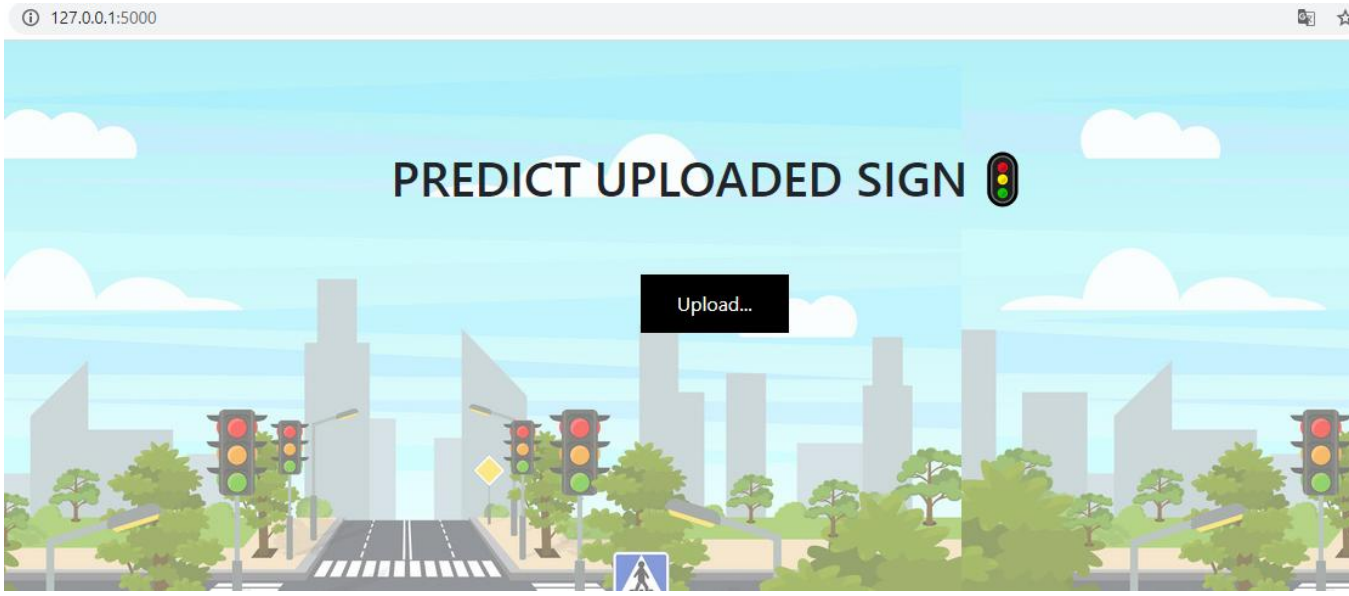


Fig 15. App running on local host

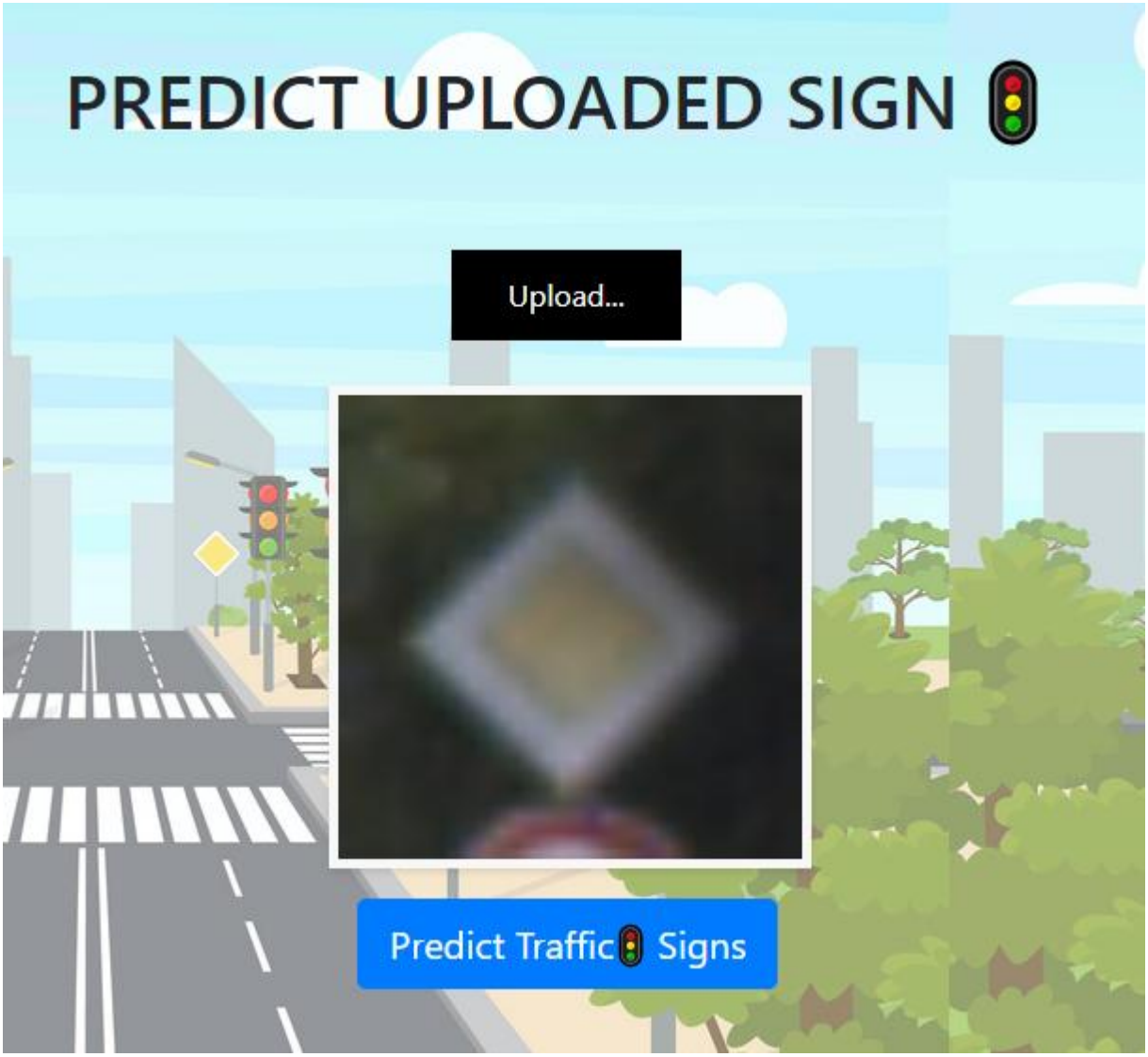
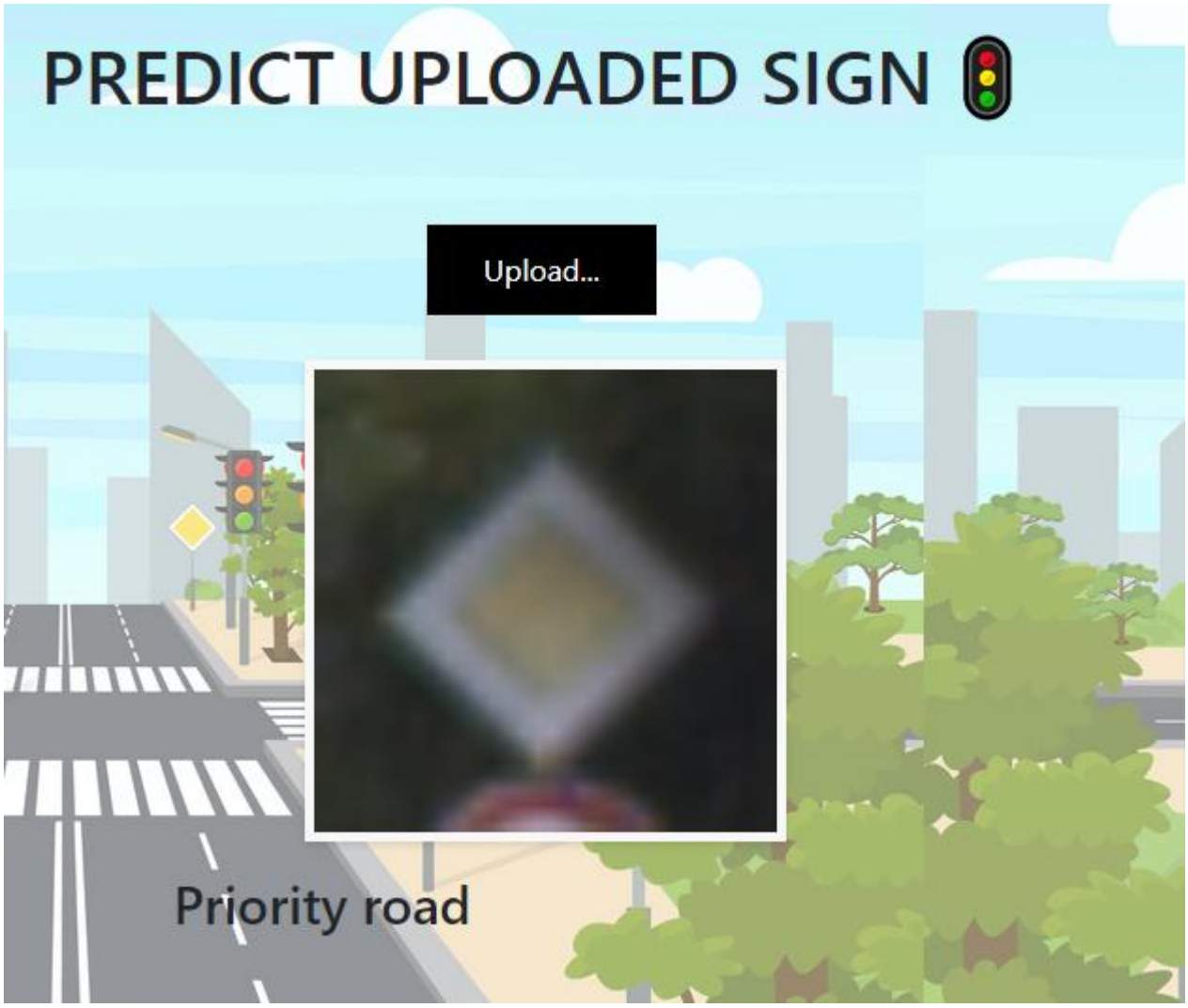


Fig 16. On uploading image


PREDICT UPLOADED SIGN
Upload...
Priority road

6. CONCLUSION

6.1 CONCLUSION:

The model accomplishes 99% precision on the approval set and 98% on the testing set. Likewise, the model makes some handling memories of about 0.1s on 4-center computer chip with no GPU speed increase. We obtained balance between content and style. We were successfully able to produce amazing results blending two photos using the technology of neural style transfer.

This task demonstrates that using deep learning to solve picture characterization is an iterative process in which it is important to first create a good pattern and then incrementally improve the model. Picture preprocessing techniques such as CLAHE, which improves contrast/splendor and increases picture information to represent real-world circumstances, helped the model to be more summarised.

6.2 Future Scope

This role may also be extended to provide a system for locating posts. A system that can continuously capture images from a camera and recognise and interpret traffic signals would be an undeniable independent framework. Further, model engineering can in any case be changed to improve execution and there is extension for most recent structures, for example, ResNet and GoogLeNet to be consolidated.

REFERENCES

[1] https://core.ac.uk/reader/29400760

[2] https://www.researchgate.net/profile/Sebastian-Houben/publication/242346625_Detection_of_Traffic_Signs_in_Real-World_Images_The_German_Traffic_Sign_Detection_Benchmark/links/0046352a03ec384e97000000/Detection-of-Traffic-Signs-in-Real-World-Images-The-German-Traffic-Sign-Detection-Benchmark.pdf

[3] https://medium.com/analytics-vidhya/german-traffic-sign-recognition-benchmark-5477ca13daa0

[4] https://towardsdatascience.com/building-a-road-sign-classifier-in-keras-764df99fdd6a

[5] https://www.youtube.com/watch?v=izK7In-WLnI&t=77s

[6] https://www.w3schools.com/howto/howto_html_file_upload_button.asp

[7] https://kanchanardj.medium.com/how-to-add-images-to-html-in-a-flask-app-4dbcc92e3aeb#:~:text=When%20we%20want%20to%20store,static%20folder%2C%20store%20the%20image.

[8] https://stackoverflow.com/questions/28207761/where-does-flask-look-for-image-files

[9] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html\

[10] http://building%20a%20very%20powerful%20image%20classification%20model%20with%20very%20little%20data/

[11] https://udai.gitbook.io/practical-ml/nn/training-and-debugging-of-nn

[12] https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/

[13] https://www.geeksforgeeks.org/graph-plotting-in-python-set-1/