Image Classification of Road Traffic Signs

1. Abstract

The main objective of this report is showcase the study behind discovering the best fitting machine learning model which can successfully classify images based on two factors, namely shape and labels. Three different ML model development technique were employed : MLP, CNN . CNN model proved to be highly accurate .Analysing of intra variability of classes and comparison of classification performance of four Convolutional Neural Networks based on parameter tuning gave us the optimal model. The images provided are grey scaled and sized to a 28*28 density. The development of the model has three stages: image preprocessing, detection, and recognition. The model demonstrated a promising result with an accuracy of $81\ \%$for shape classification an 62% for label classification. Independent evaluation was done by collecting images from internet source and real time clicks of victorian streets.

2. Introduction

Road traffic sign classification systems have multiple impact on large domains like autonomous driving , advanced driver assistance systems [1].The fundamental objective of a road sign is to provide us with useful traffic rules information. The standard computer vision techniques used could not succeed mainly due to the requirement of huge manual work [2]. Deep learning was employed to mitigate this shortcoming. The idea behind deep learning, or artificial neural networks, exists in computer science since the 1940s [3] but only with computing power today's computers can also establish themselves outside of research. The deep learning model analyse the logic structure of data similar to human perception. Neural network is the layered structured algorithm that facilitates deep learning. The data set provided is a modified version of the Belgium Traffic Sign Classification Benchmark , containing 3699 traffic sign images.

3. Dataset

The data is given in the.form of grey scale images which are 28x28 resized uniformly. The categories are defined in the form of sub directory names with sign-shape making the sub directory and sign-type making the sub-sub directory. The inherent complexity is comparatively less due to the grey scaling, since its on a single layer scale (0-255) compared to colour which has 3 layers, also we are looking for patterns in classification and properties like form, shape etc, grey level images are sufficient. On preliminary analysis few concerns were raised, which are as follow.

Concern 1 : The spread of the data with regards to shape was done to find a large weightage to round followed by triangle and least hex shaped signs, as shown in Fig 1.1 and Fig 1.2. Applying same weighing technique over labels we saw an uneven weightage trend there too with maximum number of images available for “warnings”, followed by “noentry” as shown in Fig 2.1 and Fig 2.2. This variability could lead to improper training of model which could give biased outcomes towards image category that are higher in number.

Concern 2 : Random selection of images from the pool showed distorted images from the same pool category, The reason for this mis recognition when applying Image analysis techniques is due to the peculiarity of images that are analysed or because of the distortion caused due to ageing of physical road signs or act of vandalism and even noise objects obstructing the display of road sign. As seen in Fig 3 the stop sign contains blurred as well as noise from what looks like leaves that occlude it. The “bicycle” and “trafficdirective” both contains bicycle symbol and both falls to round category. The “continue” and “traveldirectionm” contains arrows shapes of similar structural orientation.

Concern 3: some images are false categorised into wrong directories, for example “continue” directory consists of “oneway” image.This can cause false categorisation.

4. Methodology

Neural Network - In theory we can use neural network for classifying our road signals to different shapes and functional categories. The neural network present in multilayer perception constitute multiple layers like the input signals, hidden layer and the output neurons. Fig 4 shows the structure of a MLP for our dataset. Note there is no communication between neurons of the same layer. Here the neuron to neuron connection is based on weights. The first index is pointing to links beginning and the second one to the end. The loss function is calculated and back propagate this value to learn the weights metrics w1, w2. Vanishing gradient is the problem but degree is bit minimal here due to the presence of just one hidden layer. During the shape-type training using this method huge number of weights need to be trained and thats not conventional. We use the same weight as the filter used is constant

$$\mathcal{L}\left(\hat{y}^{(i)}, y^{(i)}\right) = -\sum_{j=0}^{c} y_j^{(i)} \log\left(\hat{y}_j^{(i)}\right) = -\sum_{j=0}^{c} y_j^{(i)} \log\left(o_j^{(i)}\right)$$

Convolution Neural Network (CNN) - is a deep net structure, and here we are gonna imbed learning properties to our kernel through back propagation and the kernels are 3D in nature . In our scenario we need to reduce the dimensionality of road sign image to manageable scale by removing the unwanted feature in image using feature extractor, as seen in images of “stop” presence of noise . The tenses we operate on here is the width the height and the channels of the images, which is 28*28 . Since the images are grayscale, theoretically we have only one channel but technically we have to work around it to include all 3 channels and filter out in later part, this is since the kernel used for learned filter also needs to have same number of channels as the image and here we have it as three.

Kernal Function - The kernel moves along only xy direction and provides us with feature channel also called feature map. The feature map we receive is equal to the number of kernels we apply and here we are applying multiple kernels. The kernels dimensionality is usually odd number since we want to find the middle number for feature map.Multiple convolutional layers effectively construct more complex features. The weights keep changing for each convolutional layer and we save the weights of each model so that we can start from the same place when trying to configure new configuration. We need to apply non linearity while doing convolution here and stacking the weights since convolution operation in itself is a linear function , so to make decision boundary’s non linear we have to apply non linearity activation function.

Activation Function - They are used to propogate the errors efficiently and here three different activation functions (Sigma, Than and Relu) are plotted , and Fig 5 shows the rectified non linear function which gave better results. For every change in input there is a different output for the neuron. Both the linear parts of the relu combines to produce a non linear effect.MaxPooling used will select the maximum value from the 2 dimensional region, thus reducing the dimention.

5. Modelling

On intensive research the perfect method was found to be Yang et al. [4] as it just not classified the corresponding traffic sign classes but also successfully categorised their superclass categories. Their system had 4-class SVM classifiers and the kernel used was RBF with colour Hog feature responsible for finding traffic sign categories, also real time traffic sign regognition was done using CNN which had two convolutional layer and followed by sub sampling layer.The last two layers contained a fully connected MLP. We are going to try different models to bring in maximum accuracy for classification.

Task 1 : Classifying images according to sign-shape :

In the initial approach neural network is used to predict the shape of the image. The necessary packages were installed and loaded. To unzip the file, we used zip file command in the terminal. A data frame is created to take all the shape category into consideration.Its done by iterating through the superclass of sub directory and importing raw data into shape_data, and chisel out the sign-shape features out.

Checking data uniformity - To check the uniformity of data we first group the data using sign-shape using groupby() function and plot a bar graph, to display the spread of data. The plot showed highly uneven data which will impact our model training and the performance measure used to check the performance of the model needs to mitigate this issue, we check for the f1 score for that matter.

Random images are pulled from the data by iterating through them and displayed using the random.choice() function to choose the images from random order. Data frame in spliced using k means clustering technique to provide for training and testing data and re divided to provide for validation and training set. The train size is evaluated to 2367 and test data size as 592. In the data provided we have 28x28 features and five classes - shapes , so the input layer has 784 units and the input shape is denoted by (28, 28, 3), and correspondingly the output layer has 5 units thus we provide Dense() with value 5 in output layer. The middle layer is configured to basic sigmoid activation function. we are using categorical_crossentropy because the output variable contains more than two categories. The training process can be very slow if we keep the default pixel value thus to bring it in the form range of 0-1we rescale the value by dividing it with 255, which is the maximum pixel range. The model is trained and the residual plot for loss showed a initial convergence and validation running lower to train, thus proving the train and validation are providing different results. Doing the prediction and charting a confusion matrix provides us with an fi-score accuracy of 96% and confusion matrix showed many incorrect trained objects .

Regularisation - Thus we discard above model and ways to fix it could be regularisation, in which we create a ridge penalty . The main idea is to decrease the parameters value which in turn will reduce the variance. But doing L2 regularization just made the model perform even worse and in our accuracy plot we had the validation running all over the train data line. The fi-score accuracy dropped to 88 %. We extract the hidden features by interating over each batch and a input embedding and hidden layer embedding scatter plot is created for the same. Here split apart among the scatter points shows the incapacity of the model.

Dropout - We use dropout so that neurons do not learn values of our **input.in** the training stage for each input to the network , a percentage of each neurons in hidden layer is randomly deactivated. This is done on the basis of previously defined discard probability.this is different in each layer or similar for all. Going with dropout still didn't provide for our train and validation to go along in the plot. The f1-score accuracy showed an accuracy of 97% , and the confusion matrix showed minimal false values.

CNN Model

The Fig 6 shows the execution lane of a CNN model [7]. We are mainly using kernel 5*5 and 3*3 kernels. The CNN model was build using four main building blocks :Convolutions (2D) , Pooling (or sub-sampling) ,Non-Linearity (Activations) ,Fully Connected MLP .

Configuring CNN model : In "model_cnn.add(Input(shape=(28, 28, 3))), We are doing the 2d conventional method "model_cnn.add(Conv2D(32, (3, 3) " which denotes we are using 32 convolutional kernels of 3 *3 matrix., and the inputs to this is given in "shape=(28, 28, 3)" where 28 is height and 28 is the width of the image and 3 I the channel, here though image is grayscale, still we choose 3 for a work around, for the first convolution we mention the channels, after which kenos pre determines the channels. Since we have overfitting we use weight regularisations "kernel_regularizer=regularizers.l2(0.001)" , and this is fitted into each layer the lambda term is 0.001, its applied to weights only in the first convolution layer.We using rectified liner as our activation method "Activation('relu')" .We doing a 2d pooling MaxPooling2D, in which we take the first region part, take the max value from it and then move on to next region part of the traffic signal image.In the second layer convolution we do not do any pooling , but do a 32 convolution kernel mapping and note the kernel channel input received for this layer is 32 from the previous kernel selection.

In MLP We do “model_cnn.add(Flatten())” to convert a 3d feature map to a 2d feature map. For the inner neutrons we do a rectified linear activation but for outer neurons we use softmax. The “Dropout(0.5)” used is a form of regularisation . Now we apply our dense layer, one with 64 neurons and other with 5 neurons since we have a 5 shape class classification problem here.

Optimisation mechanism -Here we using stochastic gradient descent “optimizers.SGD()”Finally we do the compilation initialised by “compile()” . This will initialise the weights for the layers that requires weights . In SDG we initialise the weights by assigning it to random small value , so compile() will first create this initialise the computational graph and also it attaches our optimisation technique, here we use “categorical_crossentropy” as our loss function . The accuracy measure chosen here is “ categorical_accuracy” , also F1 Score is the valid accuracy measure here.

Parameter Tuning : The feature map generated by CNN model has many tuneable parameters. Tuning these parameters should provide optimum accuracy model.

- Size: model_cnn3 used 5*5 as the filter size and model_cnn used 3*3 filter, here the 3*3 size filter with 50 epoch out performed the 5*5 filter model (model_cnn3)
- Depth: Depth is determined by number of filters in the convolution, here in model_cnn2 there are 3 depths while model_cnn has 2 . Again the model”_cnn outperforms model_cnn2.
- Stride: It is the distance to move the input over the centre of each filer. Since the image class value is same we do not see the requirement. Default value is 1 .
- Activation Function : the most commonly used activation function is “relu”, although model_cnn4, uses “ELU” function, and looking at res plots we could judge its not optimum for this model, thus model_cnn outperforms model_cnn4.
- Convolutional kernel value - we can choose values oof 16,32 or 64 as the kernel size, a parameter tune of 64 in model_cnn1 did not outperform did not out perform model_cnn.

Fig 7.1 shows the result and error loss for the training and accuracy data for model_cnn model and how validation and train data are in smooth sync. The confusion matrix also shows good result with minimal false positive. Fig 7.2 shows the performance metrics of the proposed model

Task 2 : Classifying images according to sign-type:

In the initial approach neural is used to predict the shape of the image. The necessary packages were installed and loaded. A data frame is created to take all the type category into consideration and called as label.

Checking data uniformity - To check the uniformity of data we first group the data using sign-Type using groupby() function and plot a bar graph, to display the spread of data. The plot extensively shows wide variation in data distribution as shown in fig 2.1 with huge weightage pertaining to “warning” symbol.

The basic operation to split the data into test train and validation data is done and data is fed into the neural network model. The output density is changed to 16 since there are 16 categories present. We will have a same input value of 28*28 since images scale remains same so input layer has 784 units and the input shape is denoted by (28, 28, 3), and correspondingly the output layer has 16 units thus we provide Dense() with value 16 in output layer. The model is trained and the residual plot for loss showed no convergence for train and validation, thus proving the train and validation are providing different results. Doing the prediction and charting a confusion matrix provides us with an fi-score accuracy of 0.97 and confusion matrix showed many incorrect trained objects. The filters are static in nature unlike CNN where kernels learn from each instances and updates there configuration.

CNN Model

The CNN model is used since weights to be calculated is huge and here we use the 5*5 and 3*3 kernel sizes and variations are produced in all the four major building blocks of CNN to fine tune the best model.

- Convolutions - the convolution is configured and it moves only in the xy directions. The input given are the grey scaled images which are 28*28 scalled in size. We have two variations to produce here, one is the size of the kernel and second with the dimension of filter.

-Initially we produced a model with 3 depth layers and first convolution with the size of 64 . The output plot portrait a validation line overlapping in a zig zag form over the train, thus its far from sync.

- A layer is removed and second layer is configured for 64 sized convolution . Size of the kernel is kept constant at 3*3, the resulting plot showed a good convergence between the validation and train but the initial positioning seems to be unmatched, this could be the scenario when model over fits with data from validation and then learns from the input keys itself.

- keeping the two layer config a 5*5 kernel is used now over the first layer, it provides better result with a f1 - score of .99 accuracy ,

- Applying regularisation to this layer to sustain over fitting, we provided a regularisation function to layer 1 of the model to Deal with over fitting nature of the model. The out put provides a more disturbed model of the lot and validation data is affected of it.

-Pooling - This will reduce the feature map size , each time a pooling happens the feature map becomes smaller and smaller until a flattening happens categorisation is reached . This change in pooling value did not yield a desirable result, in fact had a validation line hovering over the test line,

-Non - linearity - relu function is the most default function used, change of relu to sigmoid function yield a error rate because in sigmoid function for two different inputs a single output is possible, thus it dent help in inbedding non linearity.

6. Ultimate Judgement

Using CNN we can find better solution because when we use the convolution layers, it reduces the number of weights significantly also using methods like Rectified linear units which pass on errors more efficiently. The collection of data for independent evaluation was done both partially by download from google images and by clicking physical pictures on victorian streets. The main challenges was the difference in signs for same functions across countrys, also the images in google had mostly watermark imprinted on them , adding up to noise, the cropping of images to get the sign alone was a task since a lot of noise too gets add on when we try to maintain proportion. The conversion of images into a common format also was a task,

For shapes : We found the best model to be CNN model with configuration of two Layers with 32 sized convolution with kernels 3*3 in dimension . Non linearity is embedded by “relu” , we use softmax for flattening with no regularisation or drop . A simple model proved to be most efficient model giving 81 % weighted average and f1-score accuracy index.

For Types : We found the best model to be a CNN model with configuration of three layers with first layer having 64 sized convolution with kernels of 3*3 , second and third layer with 32 sized kernels with 3*3 kernels. Optimizer is SGD and relu is the embedded for non linearity, also softmax

for flattening. The model gave a 98% f1-score accuracy index while training and predicting using test data as shown in fig 8.1 and 8.2. The independent evaluation gave a result of 63% fi-score index.

7. References

[1] R. Timofte, V. A. Prisacariu, L. J. Van Gool, and I. Reid, “Chapter 3.5: Combining traffic sign detection with 3d tracking towards better driver assistance,” in Emerging Topics in Computer Vision and its Applications,C.H.Chen,Ed. WorldScientificPublishing,September 2011.

[2] Wali, S. B., Abdullah, M. A., Hannan, M. A., Hussain, A., Samad, S. A., Ker, P. J., & Mansor, M. B. (2019, May 6). Vision-Based Traffic Sign Detection and Recognition Systems: Current Trends and Challenges. Retrieved June 1, 2020, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6539654/

[3] David Kriesel. A brief overview of neural networks. 2007. url: http: // www.dkriesel.com.

[4] Y. Yang, H. Luo, H. Xu, and F. Wu, “Towards real-time traffic sign detection and classification,” IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 7, pp. 2022–2031, 2016.

[5] Editor. 2020. *Regression Analysis: How Do I Interpret R-Squared And Assess The Goodness-Of-Fit?*. [online] Blog.minitab.com. Available at: <https://blog.minitab.com/blog/adventures-in-statistics-2/regression-analysis-how-do-i-interpret-r-squared-and-assess-the-goodness-of- fit>

[6] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell. Caffe: Con- volutional architecture for fast feature embedding. CoRR, abs/1408.5093, 2014.

[7] Sermanet, P. and LeCun, Y., 2020. *Traffic Sign Recognition With Multi-Scale Convolutional Networks*. [online] Yann.lecun.com. Available at: <http://yann.lecun.com/exdb/publis/pdf/sermanet-ijcnn-11.pdf> [Accessed 1 June 2020].

Appendices

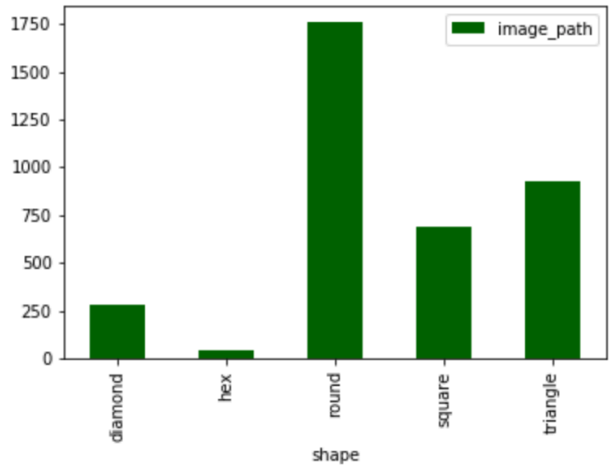


Fig 1.1

shape	image_path
diamond	282
hex	43
round	1760
square	688
triangle	926

Fig 1.2

label	image_path
bicycle	285
continue	199
crossing	95
giveway	231
laneend	118
limitedtraffic	125
noentry	375
noparking	242
parking	276
rightofway	282
roundabout	98
speed	316
stop	43
trafficdirective	195
traveldirection	124
warning	695

Fig 2.2

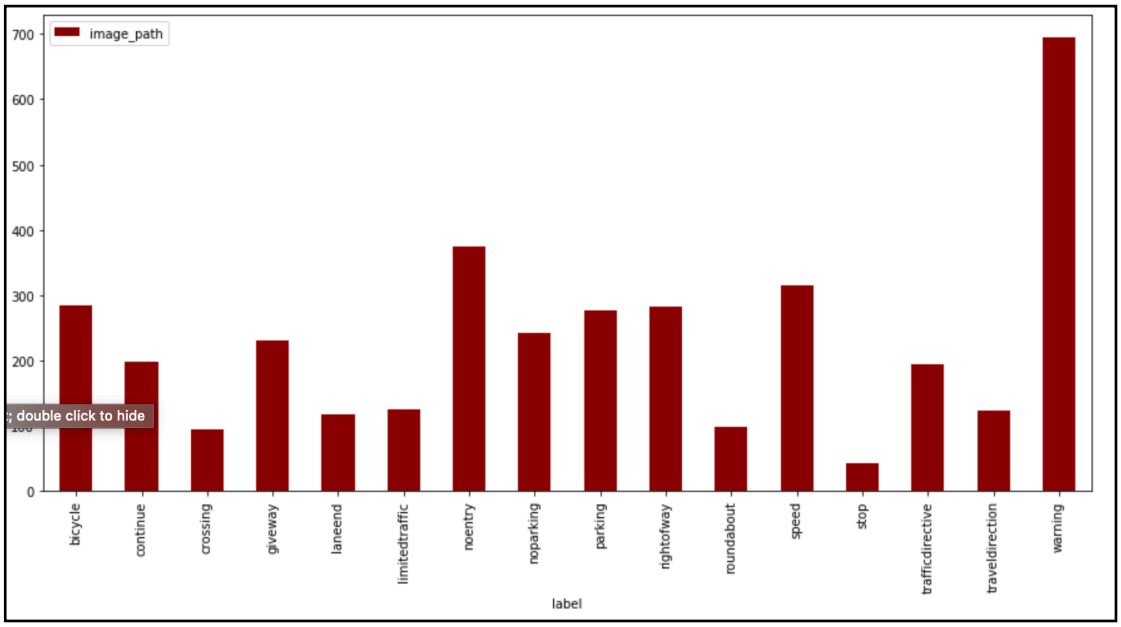
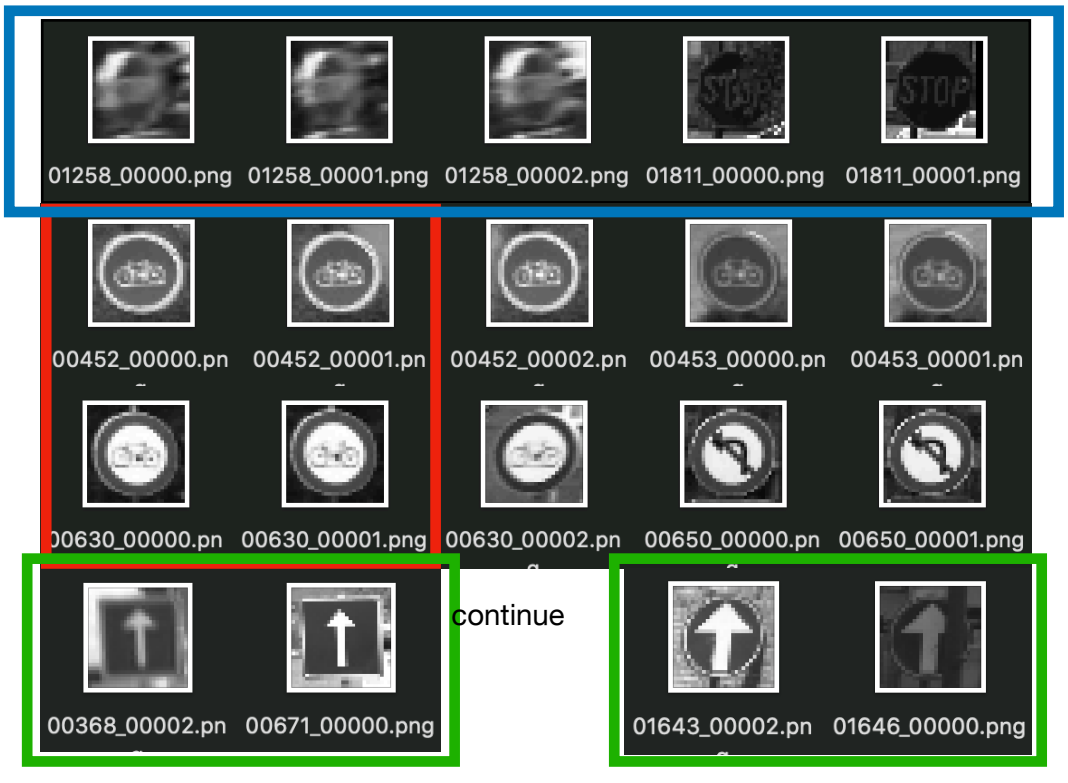


Fig 2.1



stop

bicycle

trafficdirective

continue

traveldirection

Fig 3

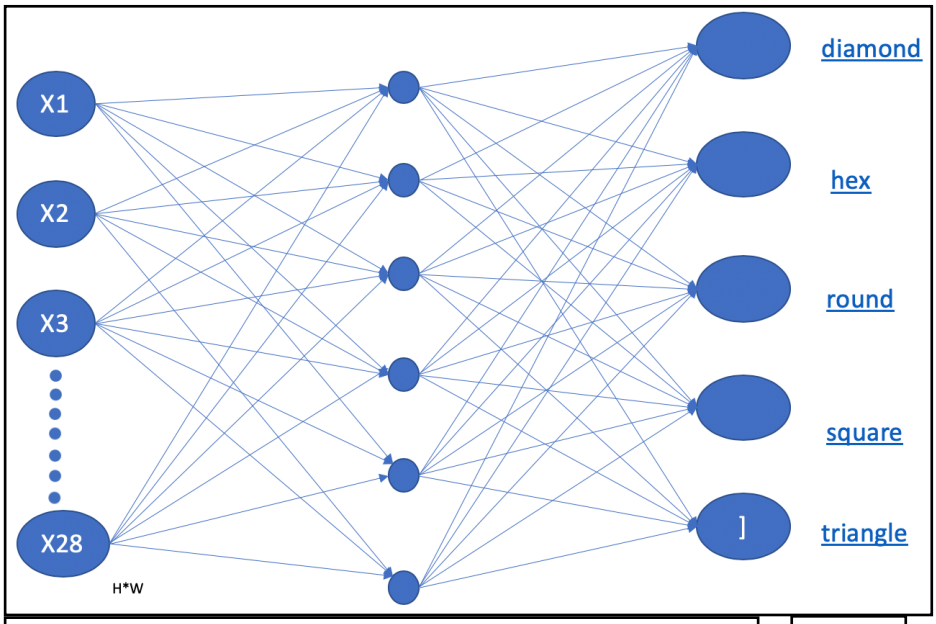


Fig 4

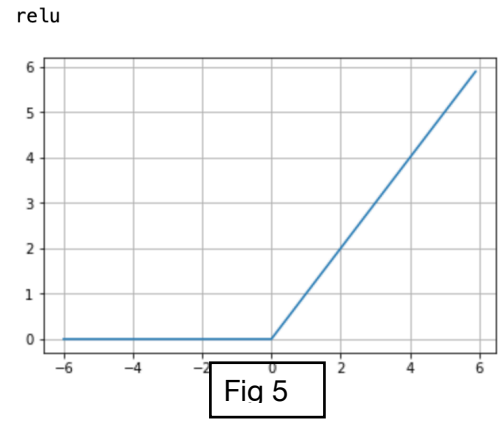


Fig 5

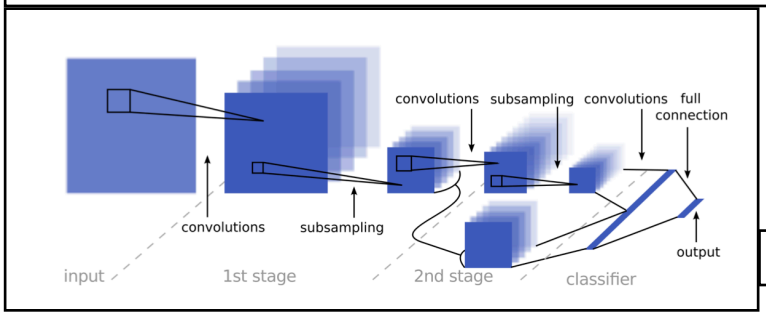
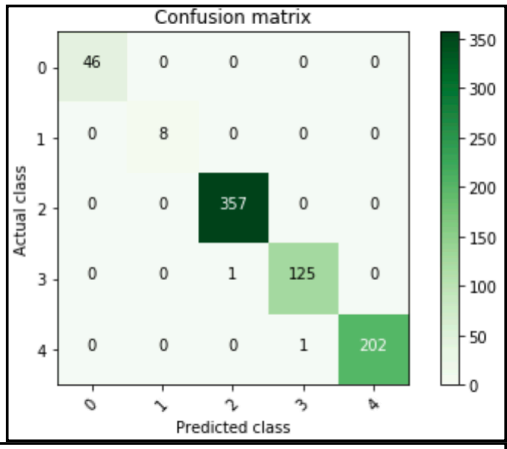


Fig 6




```
Prediction shape is (740, 5)
              precision    recall  f1-score   support

     diamond       1.00      1.00      1.00        46
         hex       1.00      1.00      1.00         8
       round       1.00      1.00      1.00       357
      square       0.99      0.99      0.99       126
    triangle       1.00      1.00      1.00       203

    accuracy                           1.00       740
   macro avg       1.00      1.00      1.00       740
weighted avg       1.00      1.00      1.00       740
```

Fig 7.2

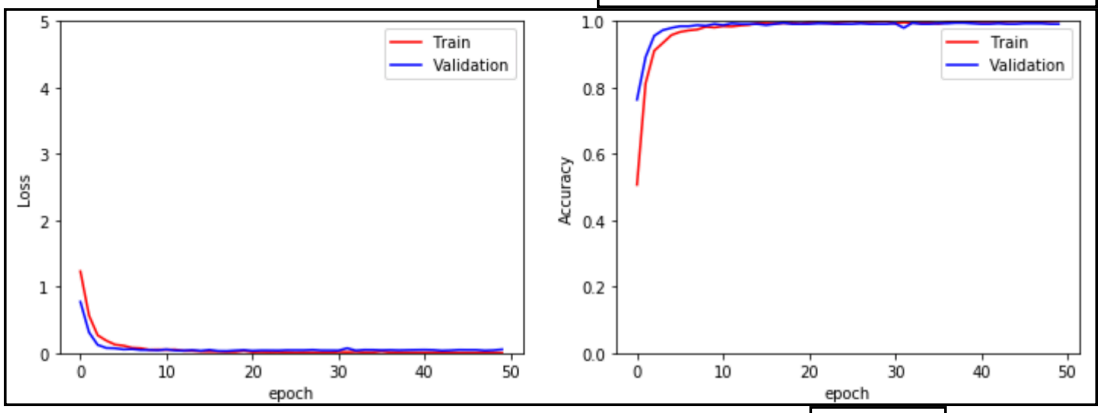


Fig 7.1

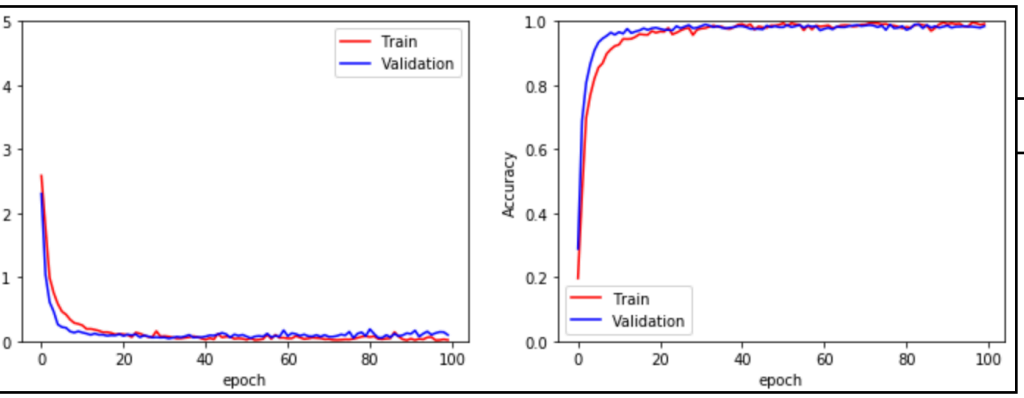
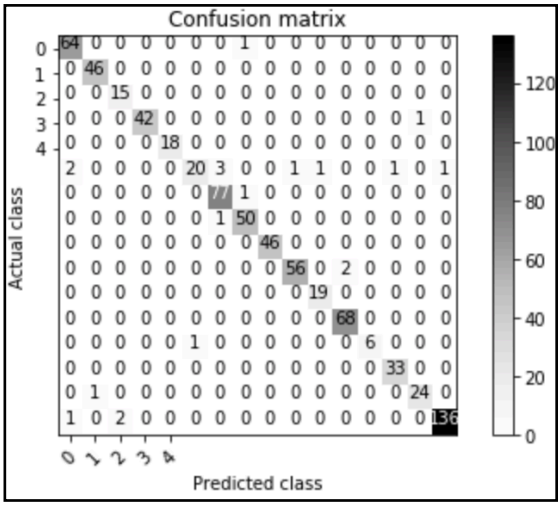


Fig 8.1




```
                precision    recall  f1-score   support

       bicycle       0.96      0.98      0.97        65
      continue       0.98      1.00      0.99        46
      crossing       0.88      1.00      0.94        15
       giveway       1.00      0.98      0.99        43
       laneend       1.00      1.00      1.00        18
limitedtraffic       0.95      0.69      0.80        29
       noentry       0.95      0.99      0.97        78
     noparking       0.96      0.98      0.97        51
       parking       1.00      1.00      1.00        46
     rightofway       0.98      0.97      0.97        58
     roundabout       0.95      1.00      0.97        19
          speed       0.97      1.00      0.99        68
           stop       1.00      0.86      0.92         7
trafficdirective       0.97      1.00      0.99        33
  traveldirection       0.96      0.96      0.96        25
        warning       0.99      0.98      0.99       139

       accuracy                           0.97       740
      macro avg       0.97      0.96      0.96       740
   weighted avg       0.97      0.97      0.97       740
```

Fig 8.2