Report on Classifying Road Traffic Signs

Author: Refeed Sultaan (s3763175)

1. Introduction

Traffic Sign Detection is one of the major problems that was introduced to Computer Vision which was later solved through Deep Learning Techniques like Convolutional Neural Networks (CNN). I was given the modified Belgium Traffic Sign dataset which contained a total of 3699 images. It contains 5 different shapes and 16 different signs (see the tables in Figure-1 for more details). There were two objectives of this project. The first objective involved detecting sign-shapes and the second task involved detecting the sign-types. Finally, I had to evaluate our best model on the independent dataset collected by me using Google Images [7] and photos taken from Australian traffic signs.

These tasks were previously solved with a feed forward Multi-Layer Perceptron Neural Network. However, the major drawbacks of MLP include redundancies in higher dimensions because all the layers are fully connected [4]. Moreover, the number of parameters grow to be a very large amount and cannot also detect patterns in an image since the images are flattened in 1 dimensional array before being processed. Multi-Layer Perceptron also needs features to be hand-engineered. On the other hand, CNN is able to find patterns inside an image by extracting features. Since layers are sparsely connected, we can increase the number of hidden layers to find extract even more features automatically. In the following sections I will discuss the methodology, where I will discuss my general approach in solving the problems related to the project.

2. Methodology

In this section, I will be mentioning the outline of my approach. Moreover, I will be discussing how I processed the images. Since there are 3699 images in the Belgium Traffic Sign dataset if I used K-Cross Validation, it would have taken as long as a week to complete all the iterations. A simple Holdout Cross Validation can capture the distribution of the original data, given that I shuffle the all images, so no ordered pattern is captured by the CNN models. All of the images were inspected to have dimensions of 28x28 pixels in Grayscale format with 1 input channel. The images were normalized to a 0-1 scale after dividing their grayscale values by 255 since the gray-scale values ranged from 0-255. The normalization ensures that input parameter has a similar distribution. Normalization was mainly done in order to make the convergence process faster while training the models [3].

After that I specified all my models in Figure 2, which will be discussed in detail in Section 3. Using those models, I would split my data into 3 sets: train set, validation set and test following the ratio 64:16:20, which will be discussed in Section 4. I will try to find the best model based on an evaluation metric (i.e. weighted f1-score) that can handle class imbalance problem which are discussed in detail in Section 4.1 and Section 4.2 on the original Belgium Traffic Sign Dataset. Later, I evaluate the best models for detecting sign-shapes and signs-types on the independent dataset to see if the models were sufficient to capture shape patterns and sign patterns inside the image in Sections 5.1 and 5.2. Finally, we make the ultimate judgement on Section 6, analysing justifying why the models work the best and discussing the limitations in our Independent Evaluation Dataset.

3. Modelling

I have chosen to use the models specified in Figure 2. CNN Architecture I have used is very similar to LeNet Architecture [6]. However, because CNN models have Dropout (0.5) implemented in them like "CNN_With_2HiddenLayers_WithDropout" model, the layers are sparsely connected allowing the model to remove redundant connections in general making it possible to detect pattern in an image unlike Multilayer Perceptron model which cannot detect patterns inside an image. CNN models are using 'Relu' activation function instead of "sigmoid" activation function, since they propagate the gradient efficiently, reducing likelihood of the vanishing gradient. The input shape of all the models were (28,28,1) fixed using a Lambda function. I used 'categorical cross entropy loss function' and soft-max classifier because there were multiple classes for each the task-1 (classifying sign-shapes) and task-2 (classifying sign-types) . For task-1 there were 5 dense nodes at the final layer because there were 5 sign-shape class labels and for task-2, there were 6 dense nodes at the final layer because there were 16 sign-type class labels. In the next section, I will be discussing hyper tuning of the models.

3.1 Hyper-tuning the models

On the models specified in Figure 2, I performed hyper tuning by changing the number of layers from 2 to 3. I could have increased the hidden layers, but since increasing layer from 2 to 3 did not have a significant increase in the performance of the model. Therefore I chose not to increase the number of hidden layers. Furthermore, I also added L1 and L2 Regularization by changing the lambda values and found 0.001 to give the best the performance for my models. I implemented early stopping to control the number of epochs based on the training loss and validation loss. When the difference of losses (i.e. min_delta) is less than 0.001, the model stops the number of epochs. I have set the patience value to 20 epochs, so the model monitors if there is a change in 'validation loss' after every 20 epochs as method to perform early stopping. While using Stochastic Gradient Optimizer (SGD) in the model I chose the learning rate to 0.01 and did not choose a smaller value because it would increase the number of epochs significantly for the training and validation loss to converge to within a 0.001 difference. Consequently, when I am running the models, I don't have to manually tune the number of epochs for each model for detecting sign-shapes and sign-types. Finally, I tested out the different optimizers e.g. 'Stochastic Gradient Descent' and 'Adam' Optimizer to see which models perform the best.

4. Evaluation

In this section, I will be discussing evaluation on the test data of the modified dataset that was provided to us. Since there was an imbalanced classes problem for both detecting sign-shapes and sign-types, a better metric to evaluate all my models is weighted F1 Score and Confusion Matrix.

Using the 'train_tests_split()' function from the scikit-learn package , I first divided dataset into train and test sets according to 80:20 ratio. Then I further split the train set into train set and validation set according to the 80:20 ratio. Since I had a large dataset, I wanted to use most of my dataset to training the model. I had 2367 train set images, 592 validation images and 740 test images. 64% of the dataset was used as train data and 16% of the data was used as

validation data 20% of the data was used as training data. I made this decision because I wanted to evaluate our data on 20% of the unseen data, to get a better evaluation and use most of our data to train the models.

4.1 Evaluation on the modified Belgium Traffic Sign Test Data for Detecting Sign-Shapes

As mentioned in Figure 1 in a table named "Modified Belgium TS data Shapes", I found the quantities for each of the respective shapes. There were 1760 images for 'round' shaped signs which was the maximum amount, whereas there were only 43 images for 'hex' shaped images which was the minimum amount. By looking at the quantities for the 5 class labels, we can clearly see that there is an imbalanced class problem. Therefore, we can conclude that accuracy is not a good metric to evaluate our data. To address the class imbalance problem, I have chosen to use weighted F1-Score and Confusion Matrix to evaluate all my models.

For detecting shape on the original Belgium traffic sign data, I found multiple models giving me a decent weighted F1-Score (as shown in Figure 3). But the best model was found to be 'CNN_With_2HiddenLayers_WithDropout ADAM_L1 Reg' which gave me the weighted F1-Score 1.00. It was able to fully recognize all the shapes in the test data. My initial thoughts were that test data of Original Belgium Traffic Sign was not representative of original data. Therefore, I checked this model on my independent evaluation traffic sign test dataset and it gave me the best result again as you will see in Section 5.1.

4.2 Evaluation on the modified Belgium Test Data for Detecting Sign-Types

From Figure 1 in a table named "Modified Belgium TS data for Signs", we can see the quantities for each of the respective signs. The maximum number of images was '695' for sign labelled 'warning' and the minimum number of images was '43' for sign labelled 'stop'. Again, by observing Figure 1 we come across the imbalanced classes problem. To mitigate problem, I had to use weighted F1-Score and Confusion matrix to evaluate all my models.

For detecting sign-types on the modified Belgium traffic sign data, I found all CNN models giving me a decent weighted F1-Score 0.97 and 0.98 (as shown in Figure 4). Consequently, I did not choose a specific model, yet at this stage. 'CNN_With_2HiddenLayers_WithDropout_ADAM' was among the candidate models that I would use for detecting sign types with a weighted F1-Score of 0.98. I was waiting for the model that would give me the best result in my Independent Evaluation Test Data.

5. Independent Evaluation

In this section, I will be discussing about the results of my independent evaluation based on the models I have. I collected 178 Traffic Sign Images using Google Images and taking screenshots of them [7] and also my taking photos of Australian Traffic Signs. All the images were RGB images. I therefore had turn them into Grayscale since our models were trained on Grayscale images. The dimensions of the images were all different and they needed to cropped and resized the images to 28x28 pixels using cv2 package in python. The quantities of the shapes and signs of the Independent Traffic Signs images can be found in Figure 1 in the tables

named “Independent TS data for Shapes” “Independent TS data for Signs”. Finally, I had normalized the images to by dividing all the pixel by 255, so that values of images remain 0 to 1, which would speed up the convergence process.

5.1 Independent Evaluation for Detecting Shapes

I used all of my 11 models as mentioned in Figure 2, to evaluate my independent set for detecting the shapes of signs. The model called ‘CNN_With_2HiddenLayers_WithDropout_ADAM_L1Reg’ as mentioned in Figure 3 performed the best with a weighted F1 Score of 0.88, which was really great. It was able to capture shapes of international traffic signs amazingly well even though the model was only trained on modified Belgium Traffic Sign Data. By looking at the confusion matrix in Figure 6 for detecting shapes on Independent Evaluation Data, it was able to correctly classify ‘diamonds’ and ‘triangle’ 100% of the time. Since the amount of hex shaped signs in the Independent Evaluation dataset was limited to only 7 images it was able to classify 5 of them correctly with a F1-Score of 0.83. However, my model was not able to capture classify the squares shapes very accurately, because of the background images of square shaped signs was not drastically very distinct than shape of the signs. This may be the reason I was able to classify 23 out of the 46 correctly, with 10 of them classified as triangle. The misclassification of square shapes into triangle signs may due to the fact some of square images were cropped poorly with only 3 edges present in some of the images. Consequently, I would resample the square images again, with better square images with all four edges showing distinctly.

5.2 Independent Evaluation for Detecting Signs

I followed the same process again, by running all of my 11 models mentioned in Figure 2 to evaluate my independent set for detecting sign patterns inside a traffic sign. The model called ‘CNN_With_2HiddenLayers_WithDropout_ADAM’, which is a 2 hidden layer CNN model with dropout and an ADAM Optimizer worked the best giving a weighted F1 score of 0.65. Belgium Traffic Signs follows the “Vienna Convention on Road Signs and Signals”. Therefore, the independent traffic sign evaluation dataset has multiple images with different kinds of signs inside them, that the model was not trained on. Therefore, it cannot properly classify the signs in the traffic sign shapes if the model has never seen those kinds of images. For example, the bicycle traffic sign in Belgium and bicycle traffic sign in another country may follow the same shape but the pattern inside the traffic sign i.e. the drawing inside the signs may be different. Since there is no international convention for traffic signs, the model was not trained on those unseen signs. This is one of the reasons I got poor results for weighted F1-Score (as shown in Figure 4).

Diving deeper into the classification report, the model was unable to classify any of the continue signs out of 5 images. I would suggest resampling the number of continue signs images with at least 50 more data ‘continue’ signs, which coincidentally was ‘square shaped’ part of badly cropped traffic signs images that might have prevented it from detecting the ‘shape’ and ‘signs’ altogether. ‘crossing’ was another squared shaped image since the backgrounds of these square images was not distinct from the shapes of the traffic signs which gave a F1-Score of 0.27. My judgement is that if the model can’t predict the shape correctly that is if the images are cropped badly with some of the edges of a sign missing in the image, it cannot predict the shape. As a result, if the shape can’t be predicted correctly, therefore the model also suffers from detecting the signs inside the image.

6. Conclusion/Ultimate Judgment

In short, my ultimate judgement is that the model ‘‘CNN With 2HiddenLayers With Dropout ADAM L1Reg’’ performed for detecting the shapes of the images and the model and “CNN With 2HiddenLayers With Dropout ADAM” performed the best for detecting sign-types There are multiple things common with both models. It is seen that CNN model works better than MLP models because MLP model can’t capture the patterns in an image because the images are flattened first. Dropout technique reduces overfitting of the models by randomly dropping redundant connections. As a result, we get better results on our test set [9]. Finally, I will be talking the advantages of ADAM Optimizer over SGD. One of the main advantages of ADAM Optimizer over Stochastic Gradient Descent Optimizer (SGD) is that ADAM Optimizer converges faster than SGD [1]. In short, this means that it would take less epochs to get the same results as SGD. ADAM has the advantages of RMS-PROP it uses squared gradient for learning rate and moving average of the gradient unlike SGD, where it uses only the gradient. For Task-1 the model we have chosen to use L1 Regularization. Regularization would reduce the chances of overfitting by adding a penalty term, causing the model to become even simpler. The advantage of L1 Regularisation over L2 Regularisation is that it causes the weights of some nodes to zero [5]. In turn, it performs a kind of feature selection by eliminating redundant connections as compared to L2 Regularisation which makes the weights non-zero. Since the problem of detecting sign-shapes is less complex than sign types, applying L1 Regularization on a CNN model with 2 convolutional hidden layers and ADAM optimizer would prevent the model from overfitting by creating a less complex model. As a result, the model would better classify on unseen independent test data. At last, following the principle of parsimony, if a 2 hidden layer CNN model solves the problem better than a 3 hidden layer CNN model, than it is better to choose a model with 2 hidden layers because it is a simpler model.

In summary, all our models work really great for test data of the original dataset, since it was basically trained on the same Belgium Traffic Sign Dataset. However, when it comes to detecting shapes of signs on Independent Evaluation Test Data. Surprisingly, our result had a high weighted F1-Score of 0.88. This may because shapes of traffic signs are usually universal. It would have increased higher, if I had access to better images of ‘continue’ and ‘crossing’ traffic signs, where the backgrounds are distinct, and all four edges of the square image are shown clearly instead of being cropped in the original image. A solution to this problem would be resampling with a better image for square images (mainly ‘continue’ and ‘crossing’) in general would boost my weighted F1-Score 0.65 to be as close to be in 0.70s range definitely.

Another major reason, of having low weighted F1-Score was that our dataset was trained Belgium Traffic Sign Dataset, which follows ‘Vienna Convention on Road Signs and Signals’ and my Independent Evaluation Dataset contained International Traffic Sign Images from United States of America, Australia, Italy and other countries. Thus, model can’t make predictions on sign-types on which it was not trained on. There would be a collection of sign-types it has never seen because signs for some sign-type, for example, ‘traffic directive’ is not universal. I will conclude by saying if we had used a 50 convolutional hidden layer architecture like ResNet-50, my score for weighted F1-Score would increase significantly, which uses concept of skip connections to make better predictions and maybe after 50 Hidden Layers, the model could find a pattern that was not visible at 2 to 3 convolutional hidden layers.

References

[1]"Adam—latest trends in deep learning optimization.", *Medium*, 2020. [Online]. Available: https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c. [Accessed: 31- May- 2020].

[2]J. Brownlee "A Gentle Introduction to Dropout for Regularizing Deep Neural Networks", *Machine Learning Mastery*, 2020. [Online]. Available: https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/. [Accessed: 31- May- 2020].

[3]"Image Data Pre-Processing for Neural Networks", *Medium*, 2020. [Online]. Available: https://becominghuman.ai/image-data-pre-processing-for-neural-networks-498289068258. [Accessed: 01- Jun- 2020].

[4]"Multilayer Perceptron (MLP) vs Convolutional Neural Network in Deep Learning", *Medium*, 2020. [Online]. Available: https://medium.com/data-science-bootcamp/multilayer-perceptron-mlp-vs-convolutional-neural-network-in-deep-learning-c890f487a8f1. [Accessed: 31- May- 2020].

[5]"Regularization Techniques | Regularization In Deep Learning", *Analytics Vidhya*, 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/. [Accessed: 31- May- 2020].

[6]M. Rizwan, "LeNet-5 - A Classic CNN Architecture - engMRK", *engMRK*, 2020. [Online]. Available: https://engmrk.com/lenet-5-a-classic-cnn-architecture/). [Accessed: 01- Jun- 2020].

[7]"traffic signs - Google Search", *Google.com*, 2020. [Online]. Available: https://www.google.com/search?hl=en&tbm=isch&sxsrf=ALeKk00o38elGfRBwUyol0rXwz3ow0xR4A%3A1590975623947&source=hp&biw=1440&bih=789&ei=h1zUXpXcN8b49QPfzbzAAg&q=traffic+signs&oq=traffic+signs&gs_lcp=CgNpbWcQAzICCAAyAggAMgIIADICCAAyAggAMgIIADICCAAyAggAMgIIADICCAA6BwgjEOoCECc6BAgjECdQ_eAKWJnzCmD8-ApoAXAAeACAAacBiAHyD5IBBDAuMTOYAQCgAQGqAQtnd3Mtd2l6LWltZ7ABCg&sclient=img&ved=0ahUKEwjVu6OHvt_pAhVGfH0KHd8mDygQ4dUDCAc&uact=5. [Accessed: 01- Jun- 2020].

[8]"Understanding and Calculating the number of Parameters in Convolution Neural Networks (CNNs)", *Medium*, 2020. [Online]. Available: https://towardsdatascience.com/understanding-and-calculating-the-number-of-parameters-in-convolution-neural-networks-cnns-fc88790d530d. [Accessed: 31- May- 2020].

[9]"Why dropouts prevent overfitting in Deep Neural Networks", *Medium*, 2020. [Online]. Available: https://medium.com/@vivek.yadav/why-dropouts-prevent-overfitting-in-deep-neural-networks-937e2543a701. [Accessed: 31- May- 2020].

Appendix

Modified Belgium TS data Shapes	
Shapes	Quantities
diamond	282
hex	43
round	1760
square	688
triangle	926

Independent TS data for Shapes	
Shapes	Quantities
diamond	11
hex	7
round	101
square	36
triangle	23

independent TS data for Signs	
Signs	Quantities
bicycle	9
continue	5
crossing	13
giveway	6
laneend	6
limitedtraffic	3
noentry	17
noparking	10
parking	12
rightofway	11
roundabout	18
speed	14
stop	7
trafficdirective	18
traveldirection	12
warning	17

Modified Belgium TS data for Signs	
Signs	Quantities
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

Figure 1: Quantities for Shapes and Signs in Both Original TS data and Independent TS data

Model	Models Type	Hidden Layer	Dropout	Regularization	Optimizer
MLP	Multilayer Perceptron	2	No	No	SGD
CNN with 2HiddenLayers	CNN	2	No	No	SGD
CNN With 2HiddenLayers WithDropout	CNN	2	0.5	No	SGD
CNN With 2HiddenLayers WithDropout ADAM	CNN	2	0.5	No	ADAM
CNN With 2HiddenLayers WithDropout ADAM L1Reg	CNN	2	0.5	L1 Regularization	ADAM
CNN With 2HiddenLayers WithDropout ADAM L2reg	CNN	2	No	L2 Regularization	ADAM
CNN with 3HiddenLayers	CNN	3	0.5	No	SGD
CNN With 3HiddenLayers WithDropout	CNN	3	0.5	No	SGD
CNN_With_3HiddenLayers_ WithDropout_ADAM	CNN	3	0.5	No	ADAM
CNN_With_3HiddenLayers WithDropout ADAM L1Reg	CNN	3	0.5	L1 Regularization	ADAM
CNN With 3HiddenLayers WithDropout ADAM L2reg	CNN	3	0.5	L2 Regularization	ADAM

Figure 2: Model Specifications used for this project

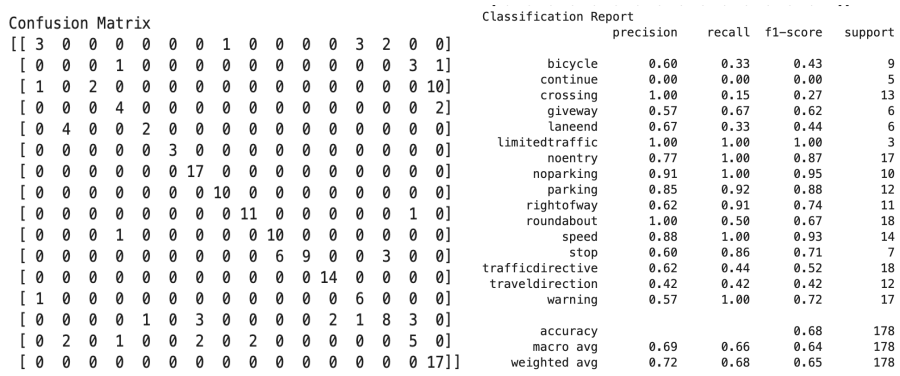
Shapes				
Model	Number of Parameters	Epoch	Modified Belgium Test Data Weighted F1-Score	Independent Test Data Weighted F1-Score
MLP	54725	99	0.96	0.5
CNN with 2HiddenLayers	61557	38	0.99	0.82
CNN With 2HiddenLayers WithDropout	61557	38	0.99	0.81
CNN With 2HiddenLayers WithDropout ADAM	61557	38	0.99	0.85
CNN With 2HiddenLayers WithDropout ADAM L1Reg	61557	82	1	0.88

CNN With 2HiddenLayers WithDropout ADAM L2reg	61557	46	0.99	0.84
CNN with 3HiddenLayers	32549	33	0.97	0.67
CNN With 3HiddenLayers WithDropout	32549	44	0.99	0.81
CNN_With_3HiddenLayers_ WithDropout_ADAM	32549	70	0.99	0.74
CNN_With_3HiddenLayers WithDropout ADAM L1Reg	32549	53	0.98	0.76
CNN With 3HiddenLayers WithDropout ADAM L2reg	32549	55	0.98	0.74

Figure 3: Evaluation on Both Original Traffic Sign Dataset and Independent Dataset for Detecting Shapes

Signs				
Model	Number of Parameters	Epoch	Modified Belgium Test Data Weighted F1-Score	Independent Test Data Weighted F1-Score
MLP	55440	109	0.92	0.29
CNN with 2HiddenLayers	61872	42	0.97	0.55
CNN With 2HiddenLayers WithDropout	61872	61	0.98	0.64
CNN With 2HiddenLayers WithDropout ADAM	61872	64	0.98	0.65
CNN With 2HiddenLayers WithDropout ADAM L1Reg	61872	45	0.98	0.58
CNN With 2HiddenLayers WithDropout ADAM L2reg	61872	64	0.98	0.63
CNN with 3HiddenLayers	33264	52	0.97	0.51
CNN With 3HiddenLayers WithDropout	33264	72	0.97	0.49
CNN_With_3HiddenLayers_ WithDropout_ADAM	33264	54	0.96	0.52
CNN_With_3HiddenLayers WithDropout ADAM L1Reg	33264	111	0.98	0.56
CNN With 3HiddenLayers WithDropout ADAM L2reg	33264	92	0.98	0.61

Figure 4:Evaluation on Both Original Traffic Sign Dataset and Independent Dataset for Detecting Signs



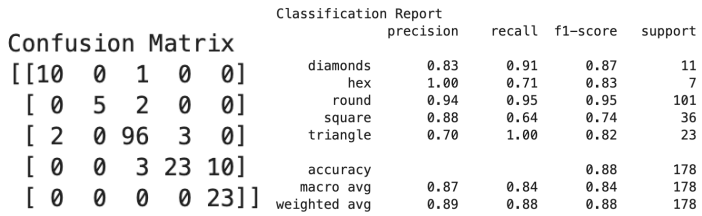
```
Confusion Matrix
[[ 3  0  0  0  0  0  0  1  0  0  0  0  3  2  0  0]
 [ 0  0  0  1  0  0  0  0  0  0  0  0  0  0  3  1]
 [ 1  0  2  0  0  0  0  0  0  0  0  0  0  0  0 10]
 [ 0  0  0  4  0  0  0  0  0  0  0  0  0  0  0  2]
 [ 0  4  0  0  2  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 17  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 10  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 11  0  0  0  0  0  1  0]
 [ 0  0  0  1  0  0  0  0  0 10  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  6  9  0  0  3  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 14  0  0  0  0  0]
 [ 1  0  0  0  0  0  0  0  0  0  0  6  0  0  0  0]
 [ 0  0  0  0  1  0  3  0  0  0  0  2  1  8  3  0]
 [ 0  2  0  1  0  0  2  0  2  0  0  0  0  0  5  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 17]]
```

```
Classification Report
                precision    recall  f1-score   support

       bicycle       0.60      0.33      0.43         9
      continue       0.00      0.00      0.00         5
      crossing       1.00      0.15      0.27        13
       giveway       0.57      0.67      0.62         6
      laneend        0.67      0.33      0.44         6
limitedtraffic       1.00      1.00      1.00         3
       noentry       0.77      1.00      0.87        17
     noparking       0.91      1.00      0.95        10
       parking       0.85      0.92      0.88        12
    rightofway       0.62      0.91      0.74        11
    roundabout       1.00      0.50      0.67        18
         speed       0.88      1.00      0.93        14
          stop       0.60      0.86      0.71         7
trafficdirective       0.62      0.44      0.52        18
 traveldirection       0.42      0.42      0.42        12
       warning       0.57      1.00      0.72        17

      accuracy                           0.68       178
     macro avg       0.69      0.66      0.64       178
  weighted avg       0.72      0.68      0.65       178
```

Figure 5: Confusion Matrix and Classification Report for independent using the best model (CNN with 2 hidden layers, dropout and adam optimizer) for sign detection



```
Confusion Matrix
[[10  0  1  0  0]
 [ 0  5  2  0  0]
 [ 2  0 96  3  0]
 [ 0  0  3 23 10]
 [ 0  0  0  0 23]]
```

```
Classification Report
              precision    recall  f1-score   support

    diamonds       0.83      0.91      0.87        11
         hex       1.00      0.71      0.83         7
       round       0.94      0.95      0.95       101
      square       0.88      0.64      0.74        36
    triangle       0.70      1.00      0.82        23

    accuracy                           0.88       178
   macro avg       0.87      0.84      0.84       178
weighted avg       0.89      0.88      0.88       178
```

Figure 6: Confusion Matrix and Classification Report for independent using the best model (CNN with 2 hidden layers, dropout, L1 Regularization and adam optimizer) for shape detection