Assignment 2- Classify Images of Road Traffic Signs

(Akshay Sachdeva – s3807399)

Perceiving traffic signs is one of the fundamental pieces of the race to extreme Self-Driving Car System and Deep Learning assumes a major job in the advancement of this framework. In this assignment I have tried various deep learning Neural Network architectures that will train on German traffic sign images to improve the street sign image classification. The task is divided into two:

1)Predicting the shape

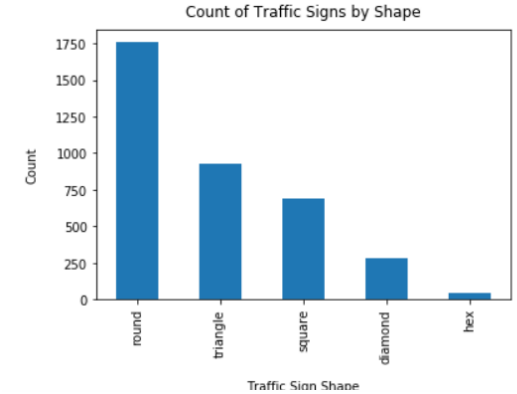
2) Predicting the sign type.

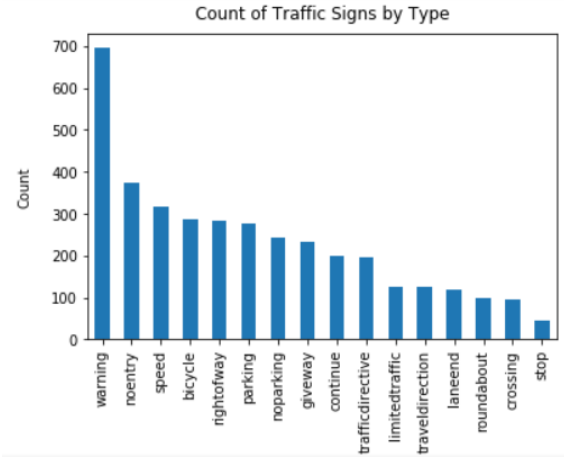
The deep learning model is built using Keras (high level API for tensorflow/ sequential processing) and I have pre-processed the images downloaded from the internet for independent evaluation using OpenCV.

Similar to any machine learning model building process we will also be executing the same golden steps defined below

1. Understand the data
2. Pre-process the data
3. Build the architecture of the model
4. Test the model
5. Iterate the same process until you achieve the optimal results

To import the data, I have used 'glob' functionality with recursive approach and created a dataframe with image path and image label. This data format is useful while using keras as we can implement ImageGenerator class to generate images and train our models.





The dataset consists of 3699 traffic sign images in .png format of varied shapes (diamond, hex, round, square and triangle) and these shapes are further subdivided into 16 sign types (stop, speed, bicycle, lane end etc.). The total images in the train set are considerably low for the given problem. It could be hard for the CNN trained on low number of images to classify real world images with high accuracy.

From the figure we can see that we have an **imbalanced class problem**. Our models could be biased towards the categories with higher count. In order to handle this problem, we could either augment the data with existing transformed images or we could use weights while training our model, i.e. give higher weights to the classes with lower count. Weight balancing balances our data by altering the *weight* that each training example carries when computing the loss. The very simple metric to measure classification is basic accuracy i.e. ratio of correct predictions to the total number of samples in dataset. However**, in the case of imbalanced classes this accuracy can be misguiding, as high metrics doesn't show prediction capacity for the minority class. F1-score** is a good choice for the imbalanced classification scenario. Along with checking the accuracy in all the

models, **I have given special consideration to the individual F1 scores to identify mis- classification of the minority class**. For this assignment I have used class weights to cater to this problem. Below are few image examples from our training set.

The data provided is in grayscale and each image corresponds to 28x28 size. I have used normalization on all the images before training. Going forward I have used test-train-split from sklearn to divide the data into train (70%) , validation(20%) and test set(10%). Next, I have created Image generators for all test/train and validation sets. I have used the batch_size as 32, which is considered a good default value as per the industry standards. Before training the algorithms all the images are normalized. Data normalization is an important step which ensures that each input parameter (pixel, in this case) has a similar data distribution. This makes convergence faster while training the network.[2]

To create the neural network architecture, I have explored both techniques i.e. function APIs and sequential processing.

Build the architecture of the model

For both tasks, the following approach was taken to identify the appropriate model for both the classification problem.

First 10 Base model architecture used:

Model No.	Hidden Layer	Activation	regularization	Drop out	optimizer(learning rate)	Batch size	epochs	weighted	Early Stopping	Overfitting	Test Accuracy	Independent evaluation Accuracy
1	1 layer	sigmoid/Softmax			sgd	32	50-100	no	No	YES	0.96	0.53
2	1 layer	relu/Softmax			sgd	32	50-100	no	No	YES	0.98	0.55
3	1 layer	relu/Softmax	L2		sgd	32	50-100	no	No	No	0.94	0.57
4	1 layer	relu/Softmax		drop(0.05)	sgd	32	50-100	no	No	No	0.98	0.57
5	1 layer	relu/Softmax		drop(0.05)	adam	32	50-100	no	No	No	0.96	0.57
6	1 layer	sigmoid/Softmax			sgd	32	50-100	Yes	No	YES	0.96	0.54
7	1 layer	relu/Softmax			sgd	32	50-100	Yes	No	YES	0.98	0.55
8	1 layer	relu/Softmax	L2		sgd	32	50-100	Yes	No	No	0.92	0.58
9	1 layer	relu/Softmax		drop(0.05)	sgd	32	50-100	Yes	No	No	0.98	0.53
10	1 layer	relu/Softmax		drop(0.05)	adam	32	50-100	Yes	No	No	0.95	0.57

In the first architecture, I have defined a simple Neural Net with 1 hidden layer using sigmoid/softmax activations with stochastic gradient decent optimizer with learning rate as 0.01. Initially, epochs were trained manually, looking at the validation loss, test loss and individual F1-Scores. In the further model I have implemented early stopping criteria to tune this parameter. In the first model the F1-score for each class in test data is above 90 for all classes except, 'hex'. This could be because of the imbalance in the input data. The overall model test accuracy is 96, however the model fails to classify the images in the independent evaluation(accuracy-59). The F1-score of each individual class is considerably low. The 2nd Model architecture was created with the similar model as the first but with 'reLU' activation. The model results improved. [1] ReLU combines the benefits of a linear activation function (no vanishing gradient) while allowing for complex relationships to be modelled in the function. Unlike sigmoid, reLU is called a piecewise function, because half of the output is linear (the positive output) while the other half 0. The ReLU function is also much less computationally taxing than sigmoid.

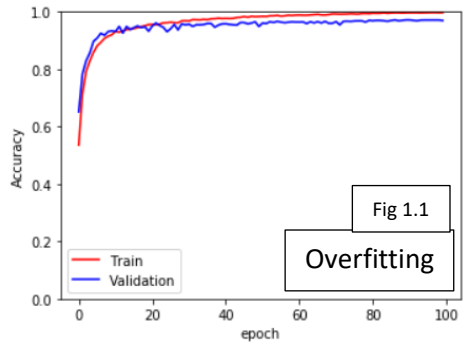


Fig 1.1

Overfitting

Looking at the validation and train error in the figure 1.1 (1st Model), we can infer that the model is overfitting for high values of epochs as the gap between train and validation accuracy increases.
In the second model, using 'reLU' activation similar overfitting problem was observed.

To cater to the overfitting there are various methods that we can use such as regularization, drop outs, data augmentation etc. Dropout is astonishing and might improve speculation of your model. Typically, you may just need to apply dropout to fully connected layers, as shared loads in convolutional layers are acceptable regularization technique themselves.
In the 3rd Model architecture (1 hidden layer, 'reLU'/'softmax' activation, sgd optimizer(lr=0.01), batch_size=32,epoch=[50,100]) , I have used regulariation in the hidden layer to reduce the overfitting.

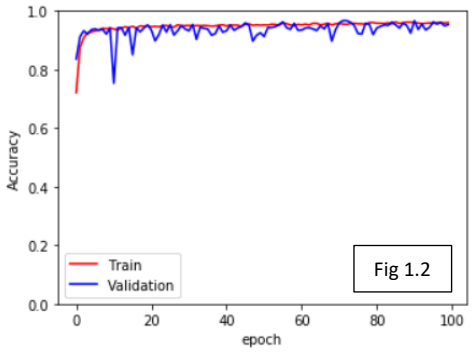


Fig 1.2

In the 3rd model, with regularization the overfitting reduced (Fig 1.2), however the overall loss increased and the model's overall F1-score reduced. Since regularization did not yield the expected results, in the next phase, instead of using regularization to treat for overfitting I have used drop outs. The results significantly improved and the test model F1-score for each class increased with overall accuracy of 98%. Also, the F1-score for each class in the independent evaluation improved.

In the 5th Model, I have tried to use a different optimizer such as 'adam'. The model does not improve from the previous model in terms of classification of the independent test set. However shows better results in terms of overfitting. As we can see in the figure-1.3 the validation accuracy is closer to that of train. In fact, the validation accuracy goes beyond the train accuracy. This could be due to training images being more complex or because of the reason that validation error is calculated at the end of epoch but the training error is calculated beginning of the epoch.

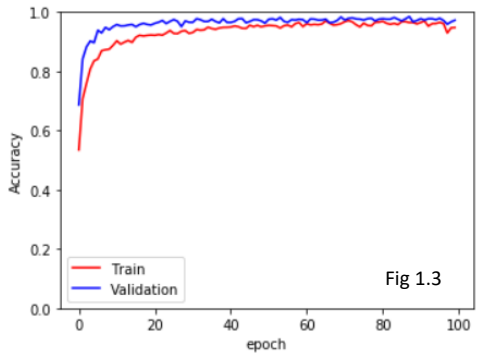


Fig 1.3

In all the above models 'hex', the class with low value was misclassified. To cater to this problem, I have used **weighted neural network**. I again ran all the above models but with higher weights to classes with a smaller number of images. It can still be observed that all the models (Model 5 to Model 10) were not able to classify images in the independent evaluation dataset with high accuracy. The models seem to be learning the exact training images and not performing well on varied data.
Going forward, I have added more complexity to the architecture and used sequential way of creating neural net architecture with 2 hidden layers. Following table shows the architectures used. All the models below were created basis the learnings from the previous models.
In Model 11, we can clearly see that as we increased the model complexity by adding more hidden layers, the F1-score improved as compared to all the pevious models. Different epochs value for each model was tried initially to check for overfitting and final evaluations. However to select the computationally optimal epoch value and avoid overfitting, I have used early stopping in the models(12,13,15 and 16).

Model No.	Hidden Layer	Activation	Pooling	Drop out	optimizer(learning rate)	Batch size	epochs	weighted	Early Stopping	Overfitting	Test Accuracy	Independent evaluation Accuracy
11	layer1	relu	Max	No	sgd	32	100-150					
	layer2	relu/Softmax	Max	No	sgd	32	100-150	No	No	Yes	0.99	0.81
12	layer1	relu	Max	No	sgd	32	250(with early stopping)					
	layer2	relu/Softmax	Max	No	sgd	32	250(with early stopping)	Yes	Yes	Yes	0.99	0.76
13	layer1	relu	Max	No	adam	32	250(with early stopping)					
	layer2	relu/Softmax	Max	No	adam	32	250(with early stopping)	Yes	Yes	Yes	1	0.8
14	layer1	relu	Max	Yes	sgd	32	100-150					
	layer2	relu/Softmax	Max	Yes	sgd	32	100-150	No	No	No	0.99	0.75
15	layer1	relu	Max	Yes	sgd	32	250(with early stopping)					
	layer2	relu/Softmax	Max	Yes	sgd	32	250(with early stopping)	Yes	Yes	No	1	0.8
16	layer1	relu	Max	Yes	adam	32	250(with early stopping)					
	layer2	relu/Softmax	Max	Yes	adam	32	250(with early stopping)	Yes	Yes	No	0.99	0.82
			Rgularization									
	layer1	relu	max	l2		32						
	layer2	relu	no	l2		32						
17	layer3	relu	max		adam	32	250(with early stopping)	Yes	Yes	No	1	0.88
	layer1	relu	max	l2		32						
	layer2	relu	no	l2		32						
18	layer3	relu	max		sgd	32	250(with early stopping)	Yes	Yes	No	0.99	0.87

Refer to this Table for reference of model Numbers used in the report for Shape classification

In Model 11-13, no drop out was used and we can clearly observe overfitting (validation accuracy decreased while the training accuracy increased) in the data as we move towards higher values of epochs. However, if compared to models 1-10, this model performs better on the test and independent evaluation set. Model 11 lacks when it comes to classifying minority classes as we can observe the corresponding F1 score is considerably low. To overcome the problem of overfitting and misclassification of minority classes, in Model 12 I have used early stopping with class weights to the model. The model improves on classifying the minority class images however overfitting still prevails in the model.

Without weights(epoch 150) (Model 11):

```
Confusion Matrix
[[15  0  3  0  0]
 [ 0  3 15  0  0]
 [ 5  0 56  5  1]
 [ 1  0  0 10  2]
 [ 2  0  2  0 25]]
Classification Report
```

- with weights(epoch 84) Model 12 (SGD)

```
Confusion Matrix
[[12  0  6  0  0]
 [ 0  7 11  0  0]
 [ 0  0 62  5  0]
 [ 0  0  0 10  3]
 [ 0  0  0  0 29]]
```

-with weights(epoch-37) Model 13 (adam)

```
Confusion Matrix
[[15  0  3  0  0]
 [ 0  9  9  0  0]
 [ 0  0 62  5  0]
 [ 0  0  0 11  2]
 [ 0  0  0  0 29]]
Classification Report
```

In the above images from models-11-13 we can clearly see, the yellow highlighted are the class ‘hex’. In the first confusion matrix we can see that only 3 were correctly classified. In model 13, with weighted architecture and adam optimizer and early stopping we were able to move closer to classifying the minority class and improving individual F1-scores.

To further explore and reduce the problem of overfitting, all the combinations in model 11-13 are replicated in model 14-15 respectively but with the addition of drop outs after each layer. It can be observed from the output that the model is not overfitting now and is able to classify on the independent test set better as compared to the previous models. In the 16th Model, I have used Adam optimizer keeping rest of the architecture same. It can be observed that using the adam optimizer the neural network requires a smaller number of epochs to converge and hence we can conclude that the adam optimizer makes the neural net faster than the sgd optimizer and therefore is computationally cheaper.

Lastly, I have tried to add another hidden layer in the architecture with max pooling including a combination of regularization and drop outs with adam optimizer. Also, to have a control over overfitting I have used early stopping as the criteria to get the computationally optimal number of epochs. I have also tried this architecture using sgd optimizer, and once again we can clearly see that

adam optimizer is more suitable to our problem and has better results in classification. In Model 17, there **is high test F1-scores for each class** and the model generalize well on the independent evaluation test set as well.

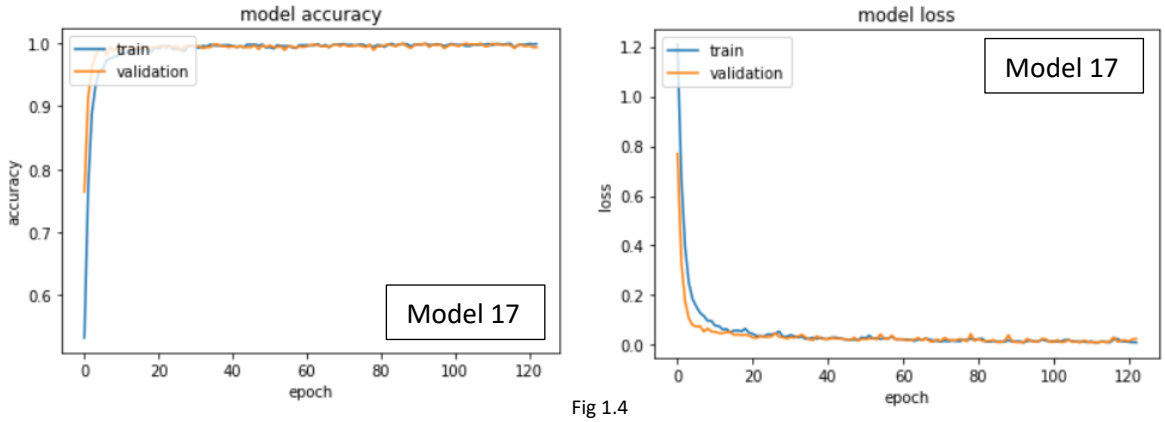


Fig 1.4

Final Judgement for shape classification:

As the Final Judgement, comparing all the neural network architectures considered to classify the sign shapes, **Model-17(3 hidden layers, with reLU/Softmax activations, pooling l2 regularization in the layers, weighted classes) appears to be performing the best with F1-score greater than 0.99 in each class on the test set**. Also, the model generalizes well when it comes to independent test evaluation. Looking at figure 1.4, we could fairly say that the model performs equally good on validation data and on the train set and does not overfit. Also, we saw while building training most of the models, overfitting is a dominant issue. Increasing layers does complicate the model, using the Lasso regularization along with drop outs, the Neural network reduces overfitting and simplifies the model to an extent with excellent classification power.

Classifying the traffic sign type

It is a bit more complex task than classifying the shape as there are a greater number of classes and each class offers further sub categories. For example, in speed sign there are various speed levels. Considering all these levels the images in the train set are considerably low for the CNN models to generalize well on the real-world images. As we know that the dataset is imbalanced, it would be hard for the learner to correctly classify the minority class images. To cater to this problem, I have used class weights, which proves to be beneficial for classifying the minority class images. As we know that the CNN is better in learning about the image outlines and shapes, it could be difficult achieving high accuracy when applied to classification of the sign type.

The same approach was followed as I did in classifying shape. All the learnings from the previous task were applied in identifying the correct traffic sign type. From the first couple of models, we observe that the models were able to learn the training images however the **individual F1-score is considerably low and the model is not able to classify the minority classes. Even applying the class weights the model perform poorly on the independent evaluation data.**

Looking at the first 10 architectures(appendix-1a), we could infer that the models are not generalizing well on the structurally different independent evaluation data. This gives an intuition that the model needs to be updated and some complex structure (increasing hidden layers) could be tried.

I have used random search to identify the model limitations and updated the model accordingly. For example, in the first few models, problem of overfitting prevails. To cater to this problem, I have tried including combinations of regularization, drop outs and early stopping. I have also used normalization

before training the models. The table in appendix 1-b include the models 11-18 that were tried. Although the table shows only accuracy, individual class F1-scores were considered to check the classification power of the model.

Final Judgement sign type classification:
The table-2a(appendix 1-b) displaying f1 scores for model 17,18, 19 and 20. All the model have almost equal accuracy on the independent evaluation test set. However, we can clearly observe from the individual F-1 Scores that model-17 performs poor in terms of the minority classes. For example, model 17 is not able to classify any sign as 'No Parking'. Therefore, we need to be careful while evaluating the models. Both model 18, 19 and 20 perform better than model 17 in terms of classifying minority classes, Model-20 being the best performing with considerably high F1 score for minority class values as well. Given the complex nature of Model 20, it is able to classify the traffic sign types better when compared to other models. The F1-score on test set is extremely good, however the overall generalization is moderate. This could be due to a lot of varied images selected in the independent evaluation dataset or due to low number of training images.

Independent evaluation set

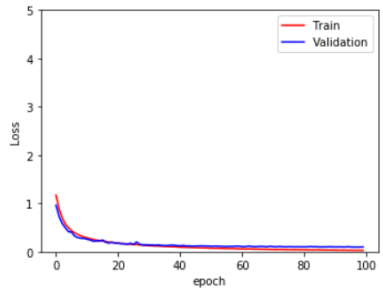
The independent evaluation set consist of 175 images with a random distribution across all shape and types. I have deliberately selected high number of 'hex' and 'diamond' images as they were in minority in the training set. Classifying these images correctly, we can assume that our model has learned well on those imbalanced classes as well. Few images are taken from the GTSRB dataset – German Traffic Sign Benchmarks, while some are taken by doing random google search. I found a smaller number of images for few sign types. Including high number of those images would have provided us a more verbose evaluation. The downloaded images were in .png format with varied shapes and sizes in rgb color. I have used OpenCV to treat these images. I rescaled these images to 28*28, grayscale as this is how our training set images were formatted. One of the limitations of the CNN model is that they cannot be trained on a different dimension of images. So, it is mandatory to have same dimension images in the dataset.

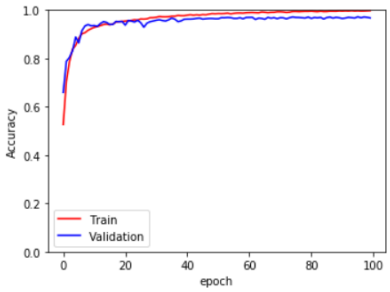
Comparing Task 1 and Task 2

In task 1- Classifying shape of the traffic sign images, I observed a considerable high accuracy and individual F1-scores for all classes in the individual evaluation set. CNN works very efficiently on classifying the shape of the traffic signs. However, when it comes to classifying the type and reading between the signs, the model is slightly less accurate as it is not able to comprehend the noise in the images. This could also be due to low number of training images. The model learns well on the training and perform good on the test set as all the images are alike, but when it comes to independent evaluation dataset the performance is moderate. The neural architecture of 3 hidden layers with reLU activation, adam optimizer, regularization/dropouts and providing class weight works best for classifying traffic sign's image shapes. The optimizer sgd performed better when we tried to predict the traffic sign type as compared to the sgd optimizer.

Appendix 1-a
Sign Shape classification

First Neural net outputs:





Confusion Matrix
```
[[ 19   0   1   0   1]
 [  0   7   2   0   0]
 [  0   0 168   1   2]
 [  1   0   1  71   2]
 [  0   0   4   0  90]]
```

test set

Classification Report

	precision	recall	f1-score	support
diamonds	0.95	0.90	0.93	21
hex	1.00	0.78	0.88	9
round	0.95	0.98	0.97	171
square	0.99	0.95	0.97	75
triangle	0.95	0.96	0.95	94
accuracy			0.96	370
macro avg	0.97	0.91	0.94	370
weighted avg	0.96	0.96	0.96	370

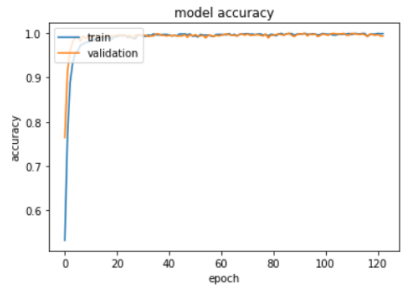
**********Independ evealuation result model
Confusion Matrix
```
[[ 8  0 16  0  0]
 [ 0  1 19  1  0]
 [ 8  0 54 19  1]
 [ 1  0  6  8  2]
 [ 0  0  8  0 21]]
```

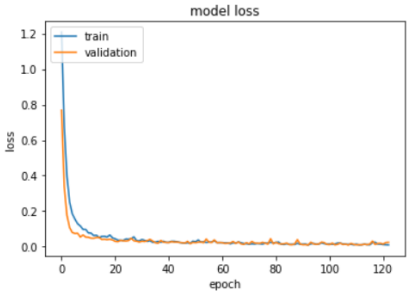
Independent evaluation test set

Classification Report

	precision	recall	f1-score	support
diamonds	0.47	0.33	0.39	24
hex	1.00	0.05	0.09	21
round	0.52	0.66	0.58	82
square	0.29	0.47	0.36	17
triangle	0.88	0.72	0.79	29
accuracy			0.53	173
macro avg	0.63	0.45	0.44	173
weighted avg	0.61	0.53	0.51	173

Final selected model-17





Confusion Matrix
```
[[ 21   0   0   0   0]
 [  0   9   0   0   0]
 [  0   1 170   0   0]
 [  0   0   0  75   0]
 [  0   0   0   0  94]]
```

test set

Classification Report

	precision	recall	f1-score	support
diamonds	1.00	1.00	1.00	21
hex	0.90	1.00	0.95	9
round	1.00	0.99	1.00	171
square	1.00	1.00	1.00	75
triangle	1.00	1.00	1.00	94
accuracy			1.00	370
macro avg	0.98	1.00	0.99	370
weighted avg	1.00	1.00	1.00	370

Confusion Matrix
```
[[21  0  2  0  1]
 [ 0 18  3  0  0]
 [ 3  0 70  7  2]
 [ 0  0  0 15  2]
 [ 0  0  0  0 29]]
```

Independent evaluation test set

Classification Report

	precision	recall	f1-score	support
diamonds	0.88	0.88	0.88	24
hex	1.00	0.86	0.92	21
round	0.93	0.85	0.89	82
square	0.68	0.88	0.77	17
triangle	0.85	1.00	0.92	29
accuracy			0.88	173
macro avg	0.87	0.89	0.88	173
weighted avg	0.90	0.88	0.89	173

APPENDIX 1-b

Sign Type classification

Model Architecture for classifying sign types

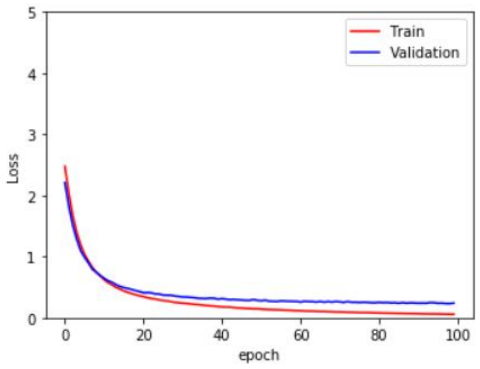
Model No.	Hidden Layer	Activation	regularization	Drop out	optimizer(learning rate)	Batch size	epochs	weighted	Early Stopping	Overfitting	Test Accuracy	Independent evaluation Accuracy
1	1 layer	sigmoid/Softmax			sgd	32	50-100	no	No	YES	0.97	0.36
2	1 layer	relu/Softmax			sgd	32	50-100	no	No	YES	0.97	0.34
3	1 layer	relu/Softmax	L2		sgd	32	50-100	no	No	No	0.86	0.32
4	1 layer	relu/Softmax		drop(0.05)	sgd	32	50-100	no	No	No	0.98	0.49
5	1 layer	relu/Softmax		drop(0.05)	adam	32	50-100	no	No	No	0.97	0.46
6	1 layer	sigmoid/Softmax			sgd	32	50-100	Yes	No	YES	0.97	0.41
7	1 layer	relu/Softmax			sgd	32	50-100	Yes	No	YES	0.97	0.33
8	1 layer	relu/Softmax	L2		sgd	32	50-100	Yes	No	No	0.92	0.42
9	1 layer	relu/Softmax		drop(0.05)	sgd	32	50-100	Yes	No	No	0.97	0.39
10	1 layer	relu/Softmax		drop(0.05)	adam	32	50-100	Yes	No	No	0.97	0.41

Model No.	Hidden Layer	Activation	Pooling	Drop out	optimizer(learning rate)	Batch size	epochs	weighted	Early Stopping	Overfitting	**Test Accuracy**	Independent evaluation Accuracy
11	layer1	relu	Max	No	sgd	32	100-150					
	layer2	relu/Softmax	Max	No	sgd	32	100-150	No	No	Yes	0.96	0.52
12	layer1	relu	Max	No	sgd	32	250(with early stopping)					
	layer2	relu/Softmax	Max	No	sgd	32	250(with early stopping)	Yes	Yes	Yes	0.98	0.49
13	layer1	relu	Max	No	adam	32	250(with early stopping)					
	layer2	relu/Softmax	Max	No	adam	32	250(with early stopping)	Yes	Yes	Yes	0.98	0.58
14	layer1	relu	Max	Yes	sgd	32	100-150					
	layer2	relu/Softmax	Max	Yes	sgd	32	100-150	No	No	No	0.99	0.58
15	layer1	relu	Max	Yes	sgd	32	250(with early stopping)					
	layer2	relu/Softmax	Max	Yes	sgd	32	250(with early stopping)	Yes	Yes	No	0.99	0.58
16	layer1	relu	Max	Yes	adam	32	250(with early stopping)					
	layer2	relu/Softmax	Max	Yes	adam	32	250(with early stopping)	Yes	Yes	No	1	0.7
			Rgularization									
	layer1	relu	max	l2		32						
	layer2	relu	no	l2		32						
17	layer3	relu	max		adam	32	250(with early stopping)	Yes	Yes	No	0.99	0.72
	layer1	relu	max	l2		32						
	layer2	relu	no	l2		32						
18	layer3	relu	max		sgd	32	250(with early stopping)	Yes	Yes	No	0.98	0.72
				dropout								
	layer1	relu+Batch Normalization	Max	0.2		32						
	layer2	relu+Batch Normalization	no			32						
	layer3	relu+Batch Normalization	Max	0.2		32						
	layer4	relu+Batch Normalization	no			32						
19	layer5	relu+Batch Normalization	no		Adam	32	250(with early stopping)	Yes	Yes	No	0.99	0.7
	layer1	relu+Batch Normalization	Max	0.2		32						
	layer2	relu+Batch Normalization	no			32						
	layer3	relu+Batch Normalization	Max	0.2		32						
	layer4	relu+Batch Normalization	no			32						
20	layer5	relu+Batch Normalization	no		sgd	32	250(with early stopping)	Yes	Yes	No	0.99	0.73

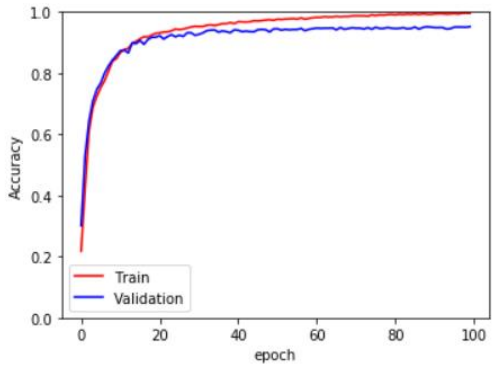
Best performing model in terms of individual F1-scores

First Base model output:

Model Loss

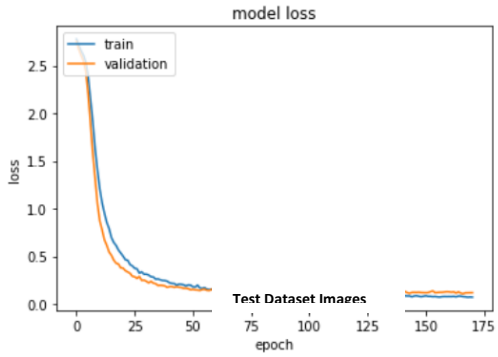


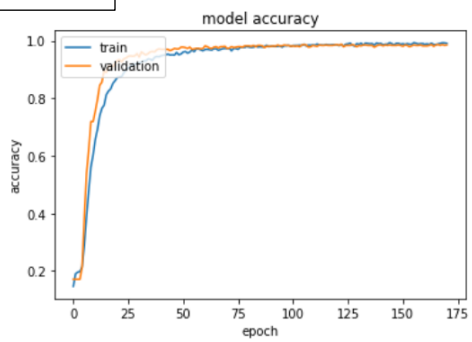
Model Accuracy



Final Selected model-20:

Model 20





Classification Report

Test set F1-scores

	precision	recall	f1-	
bicycle	1.00	1.00	1.00	28
continue	1.00	1.00	1.00	18
crossing	1.00	1.00	1.00	12
giveway	1.00	1.00	1.00	24
laneend	0.92	1.00	0.96	12
limitedtraffic	0.85	1.00	0.92	11
noentry	1.00	1.00	1.00	41
noparking	1.00	1.00	1.00	30
parking	1.00	1.00	1.00	33
rightofway	0.95	0.95	0.95	21
roundabout	1.00	1.00	1.00	5
speed	0.96	0.96	0.96	26
stop	1.00	1.00	1.00	9
trafficdirective	1.00	0.86	0.92	21
traveldirection	1.00	1.00	1.00	9
warning	0.99	0.99	0.99	70
accuracy			0.98	370
macro avg	0.98	0.98	0.98	370
weighted avg	0.98	0.98	0.98	370

F1-scores on independent evaluation dataset

Table 2a

Model 17

	precision	recall	f1-score	support
bicycle	0.33	0.14	0.20	7
continue	0.19	1.00	0.32	3
crossing	0.67	0.50	0.57	4
giveway	0.82	1.00	0.90	14
laneend	1.00	1.00	1.00	5
limitedtraffic	1.00	0.60	0.75	10
noentry	0.67	0.18	0.29	11
noparking	0.00	0.00	0.00	7
parking	0.80	0.80	0.80	5
rightofway	0.84	0.88	0.86	24
roundabout	0.82	1.00	0.90	9
speed	0.83	1.00	0.91	15
stop	0.95	0.86	0.90	21
trafficdirective	0.43	1.00	0.60	6
traveldirection	0.75	0.18	0.29	17
warning	0.68	1.00	0.81	15

Model 18

	precision	recall	f1-score	support
bicycle	0.67	0.29	0.40	7
continue	0.20	1.00	0.33	3
crossing	0.67	0.50	0.57	4
giveway	0.93	1.00	0.97	14
laneend	1.00	1.00	1.00	5
limitedtraffic	0.91	1.00	0.95	10
noentry	0.67	0.18	0.29	11
noparking	0.36	0.56	0.43	9
parking	1.00	0.80	0.89	5
rightofway	0.88	0.88	0.88	24
roundabout	1.00	0.56	0.71	9
speed	0.92	0.80	0.86	15
stop	0.95	0.86	0.90	21
trafficdirective	0.40	1.00	0.57	6
traveldirection	0.33	0.12	0.17	17
warning	0.75	1.00	0.86	15

Model 19

	precision	recall	f1-score	support
bicycle	0.75	0.43	0.55	7
continue	0.20	1.00	0.33	3
crossing	0.67	0.50	0.57	4
giveway	0.93	1.00	0.97	14
laneend	1.00	1.00	1.00	5
limitedtraffic	1.00	1.00	1.00	10
noentry	0.67	0.18	0.29	11
noparking	0.33	0.56	0.42	9
parking	0.50	0.80	0.62	5
rightofway	0.78	0.88	0.82	24
roundabout	0.64	1.00	0.78	9
speed	1.00	0.40	0.57	15
stop	1.00	0.86	0.92	21
trafficdirective	0.25	0.17	0.20	6
traveldirection	0.50	0.24	0.32	17
warning	0.75	1.00	0.86	15

Model 20

	precision	recall	f1-score	support
bicycle	0.75	0.43	0.55	7
continue	0.30	1.00	0.46	3
crossing	1.00	0.50	0.67	4
giveway	1.00	1.00	1.00	14
laneend	1.00	1.00	1.00	5
limitedtraffic	0.83	1.00	0.91	10
noentry	0.33	0.09	0.14	11
noparking	0.29	0.56	0.38	9
parking	1.00	0.80	0.89	5
rightofway	0.83	0.83	0.83	24
roundabout	1.00	0.22	0.36	9
speed	0.93	0.93	0.93	15
stop	1.00	0.86	0.92	21
trafficdirective	0.40	1.00	0.57	6
traveldirection	0.60	0.35	0.44	17
warning	0.75	1.00	0.86	15

References:

[1]https://www.quora.com/What-is-the-difference-between-sigmoid-and-ReLU [1

[2]https://becominghuman.ai/image-data-pre-processing-for-neural-networks498289068258

[3] https://keras.io/api/preprocessing/image/

[4] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, *Activation Functions: Comparison of Trends in Practice and Research for Deep Learning* , viewed on 28 May 2020, website https://arxiv.org/pdf/1811.03378.pdf

[5] Federal ministry of education and research, viewed on 26th May 2020,website: http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset

[6] V. Fung, *An Overview of ResNet and its Variants*, viewed on 26th May 2020, website https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035#:~:text=In%20this%20novel%20architecture%2C%20the,are%20aggregated%20with%20depth%2Dconcatenation.