Computational Machine Learning (COSC2763)

Assignment 2 - Project 1: Classify image of road traffic signs

Vishwa Gandhi (S3714805)

Contents

Aim

In this report, problem of classifying images of road traffic signs using supervised machine learning techniques. There are major two tasks to perform on given data which includes classifying the traffic sign-shape and classifying the traffic sign-type. The data given for the project is a modified version of Belgium Traffic Sign Classification Benchmark. This data has 28*28 grayscale images of road traffic signs taken from real-world vehicle. Project is divided in two parts: Model Analysis & Independent Evaluation.

Methodology

Data Exploration

Original data is in the 28*28 grayscale image format. All images are of the same size. There are total 5 unique sign-shape and 16 total sign-type in the dataset. Image is already in grayscale format. In our

problem, there is no major significance of color. Hence using grayscale for classification is a good idea. It will also maximize the distance between different classes.

In a classification task, distribution analysis of classes is very important. Below given figure shows the distribution of different classes for shape and type. Evidently, it can be said that count of images for shapes like hex, diamond and types like stop and crossing is comparatively lower than other attributes.

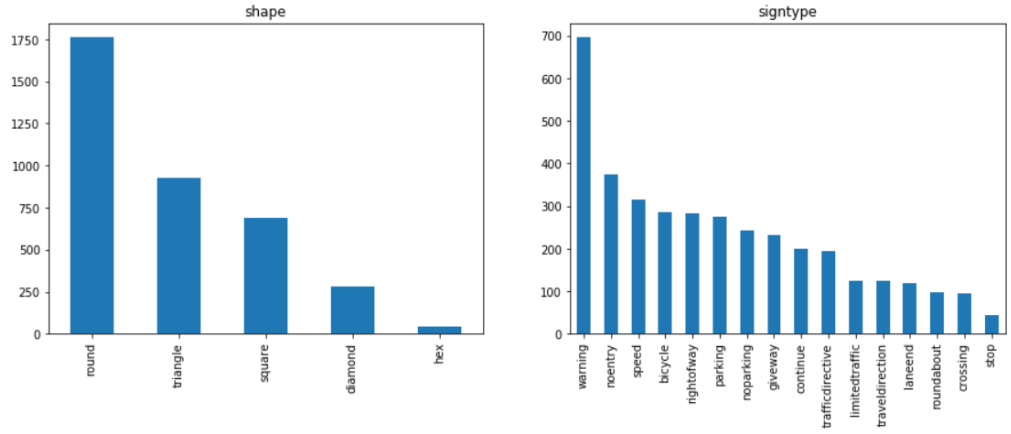


Figure 1. Image class distribution of sign shape and type

This is a limitation of the data. I have used stratified with train test split to maintain the same ratio of each class count in train and split data as original data. Also, I have augmented available images to reduce the over fitting effect. To handle this imbalance of data, extra data for minority class categories could have been generated by augmenting the given dataset images only.

Investigation of Supervised Image Classification Techniques

Supervised algorithms like SVM, Decision Tree, KNN can be used for image classification. However, these models do not properly classify the data. To accomplish image classification, it is very essential and critical to extract features from the image. Generally, above mentioned algorithms are not capable of extracting features from raw images. On the other hand, CNN extracts the features from the raw image while learning.

In case of Neural Network, parameter grows very rapidly. Tuning parameters can make model computationally heavy. CNN diminishes parameter tuning time. Another benefit of CNN over conventional supervised algorithms is its capability to reduce dimensionality. Feature engineering is completely handled by CNN.

CNN also has some other architectures like VGGNet, LeNet, GoogLeNet. All these architectures are designed around some agenda. For example, VGGNet is a very deep net and leverages performance improvement for larger and complex problem. I believe that advanced CNN architectures are not required considering provided dataset.

Here, Keeping the project complexity in the mind, I have used CNN model. My approach is to create simple and effective model for image classification.

Implementation Approach

I have implemented two types of supervised machine learning algorithm. Main algorithm for my project is Convolutional Neural Network (CNN). For initial analysis, I have used simple Neural Network model. I have explored how hidden layers work in trained NN. Then I have implemented CNN with and without augmentation of data. We will analyze how data augmentation and parameters can change the performance of model in terms of categorical loss, accuracy, f1-score, precision and recall.

ImageDataGenerator has been used to process data in batches. This will improve performance as data will only be loaded in batches into memory. It enhances performance when the dataset is large.

Neural Network:

Following points indicates my approach while selecting model configuration. This remains same for shape and sign analysis.

- There are two options to create models in 'Keras' package. It can be done by either Sequential or Functional API. I have used Functional API for NN model.
- Original images are in shape (28, 28, 1). But ImageDataGenerator automatically converts into 4-channels. Hence, Input shape (28, 28, 3) is used.
- 'Softmax' activation is used at output layer which generate list of probabilities of all possible outcomes. It is a good option for multi-class classification.
- Hidden layer has 'Sigmoid' activation applied. I have used sigmoid function because it is suitable for the model which calculates probabilities as it operates between (0, 1). This meets our need.

With configured NN model and 100 epochs, following result is achieved.

Evaluation Metric	**Sign-Shape**	**Sign-Type**
Training Categorical Accuracy	0.9996	0.9996
Validation Categorical Accuracy	0.9611	0.9578
Training loss	0.0118	0.0233
Validation loss	0.1304	0.1701
Training F1-meausre	0.9996	0.9991
Validation F1-measure	0.9609	0.9613

Also, loss curve and accuracy curve in Figure 7 and Figure 8reveals that our models are getting over-fitted to the training data set. This can be observed from the fact that training accuracy is higher than validation accuracy. Likewise, training loss is lesser than validation loss. Therefore, it can be said that training model performance is higher.

One more interesting behavior of NN model's hidden layer is plotted in Figure 9 and Figure 10for shape classification. These Figures represent the output from input layer and desired hidden layer for Training and validation data. It can be concluded from these figures that hidden layer output for training data is tightly clustered than it is for validation data. This supports the result of Loss and accuracy curves. The same kind of behavior is present for the type classification model.

Deep CNN Network

As discussed above, CNN is used for both the classification tasks to leverage image feature engineering ability of model.

Model Configuration choices involves following:

- I have already mentioned the choices I made for input size and activation function. Here, i have l2 regularization. It is reducing the impact of bias in input data. Value for l2 is set by manually experimenting parameter with values (0.001,0.01, 0.05).
- Optimizer used is Stochastic Gradient Descent called 'SGD'. SGD implements minima which prevents from the local minima. It tries to find flatter minima. There are other optimizers like 'Adam' and 'RMSProp'. There is some trade off in using other optimizers. For example, ADAM quickly converges towards sharper minima which are less desirable. But the ability to deal with local minima is the key element which made me choose SGD over other optimizers.

Hyper parameter tuning

I have performed hyper parameter tuning using KerasClassifier and RandomSearchCV. Initially I tried with GridSearchCV. But considering the complexity of the setting, I have picked RandomSearchCV. I have also used 2-fold cross validation to reduce the effect of overfitting

I have tunes CNN model nodes, epochs, learning rate and momentum to optimize the performance of the model. Using these best parameters I have trained my final models. Results are as follow.

```
Best: 0.980737 using {'output_class': 5, 'n4': 64, 'n3': 64, 'n2': 16, 'n1': 64, 'momentum_param': 0.9, 'learning_rate': 0.001,
'epochs': 150, 'batch_size': 15}
```

Figure 2 Best parameter for shape CNN

```
Best: 0.975667 using {'output_class': 16, 'n4': 32, 'n3': 64, 'n2': 64, 'n1': 64, 'momentum_param': 0.9, 'learning_rate': 0.01,
'epochs': 100, 'batch_size': 15}
```

Figure 3 Best parameter for type CNN

Model Evaluation and Ultimate Judgement

With the best parameter achieved from the hyper parameter tuning, final model for Shape and Type analysis are trained. I have used

- Sign-Shape analysis:
    - When trained a model with raw images, model on training data was slightly performing better than on validation data in terms of loss, accuracy and f1-score.

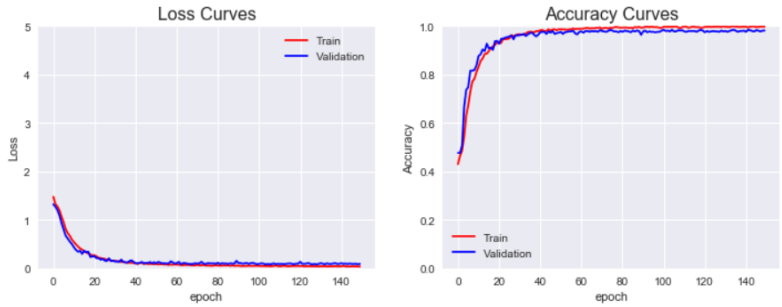
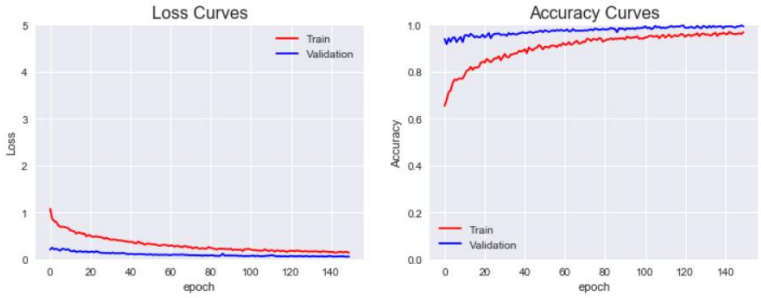


Figure 4 Trend of Shape CNN on raw images

- To identify the reason, I then trained another CNN model on augmented images. This resulted in better performance of model on validation data. Augmentation makes training data more complicated for the model to learn. Hence, it gives better model and performance on validation data. Augmentation in this key is a good technique to improvise the performance and to make the model learn more.



- Testing the final model trained on augmented data gives 0.99 f1-score and 0.99 weighted average precision.

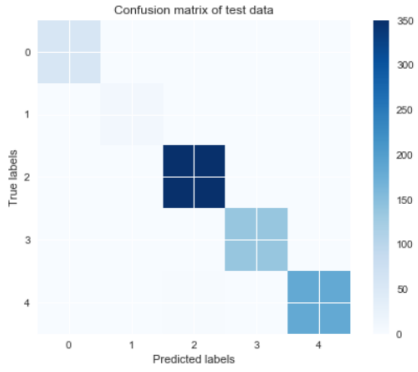


Figure 5 Confusion Matrix Shape CNN on testing data

- Sign-Type Analysis:
  - For sign type CNN model, both the models, one trained on raw images and another on augmented images, exhibits higher performance for validation data than training data. This indicated that model has learnt the weights correctly and parameters tuned are appropriate.

- Testing the augmented model on unseen test data gives 0.98 f1-score and 0.98 weighted average precision

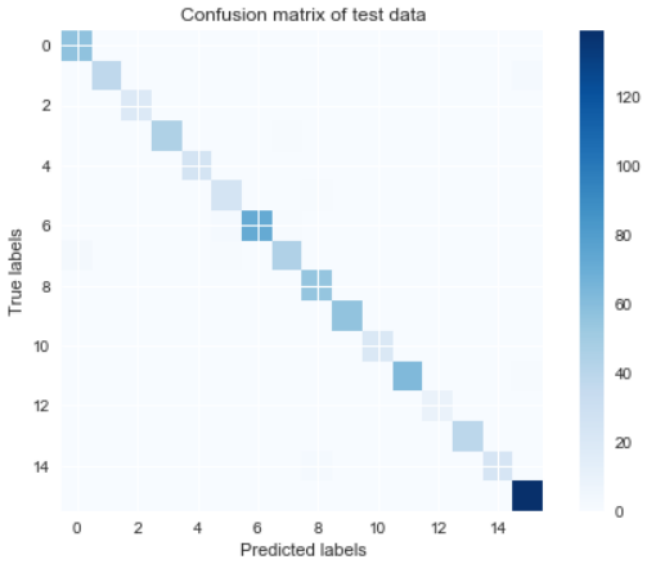


Figure 6 Confusion Matrix Type CNN on testing data

Independent Evaluation

I have acquired traffic sign data from Australia, Germany and Italy ([3] & [4]). The data is arranged according the sign type and shape labels in the original dataset. This dataset is prepared to evaluate performance on the data from other countries. This evaluation will show how model generalized to unseen data with different format.

- Shape CNN model gives 0.78 F1-score for evaluation data.
- Type CNN model gives a slightly lower F1- score of 0.55.

Hence, this model which is only trained on Belgian dataset is not appropriate for global level analysis.

I had challenges in finding and labeling appropriate data from various countries. It was a difficult task to find in valid format to feed into model.

Future Work

This analysis can be further advanced by performing computationally expensive hyper parameter tuning and bigger CNN models. As I was having limited computational resources, I had to bound the limit for experiments.

References

[1] Victor Adrian Prisacariu, Radu Timofte, Karel Zimmermann, Ian Reid, Luc van Gool, Integrating object detection with 3D tracking towards a better driver assistance system, IAPR International Conference on Pattern Recognition, ICPR 2010. [pdf][bib].

[2] Radu Timofte, Karel Zimmermann, Luc van Gool, Multi-view traffic sign detection,

recognition, and 3D localisation, IEEE Workshop on Applications of Computer Vision, WACV 2009. [pdf][bib]

[3] Btsd.ethz.ch. 2020. Belgiumts - Belgian Traffic Sign Dataset. [online] Available at: <https://btsd.ethz.ch/shareddata/> [Accessed 29 May 2020].

[4] Cvl.isy.liu.se. 2020. Download. [online] Available at: <http://www.cvl.isy.liu.se/en/research/datasets/traffic-signs-dataset/download/> [Accessed 29 May 2020].

[5] Dario, A., 2020. DITS. [online] Users.diag.uniroma1.it. Available at: <http://users.diag.uniroma1.it/bloisi/ds/dits.html> [Accessed 29 May 2020].

[6] Sid.erda.dk. 2020. Public Archive: Daaeac0d7ce1152aea9b61d9f1e19370. [online] Available at: <https://sid.erda.dk/public/archives/daaeac0d7ce1152aea9b61d9f1e19370/published-archive.html> [Accessed 29 May 2020].

Appendix

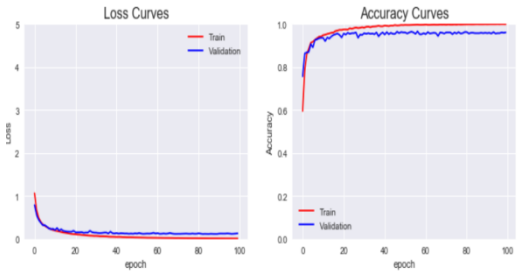


Figure 7 Shape NN Trends

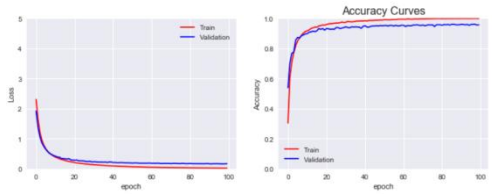


Figure 8Type NN Trends

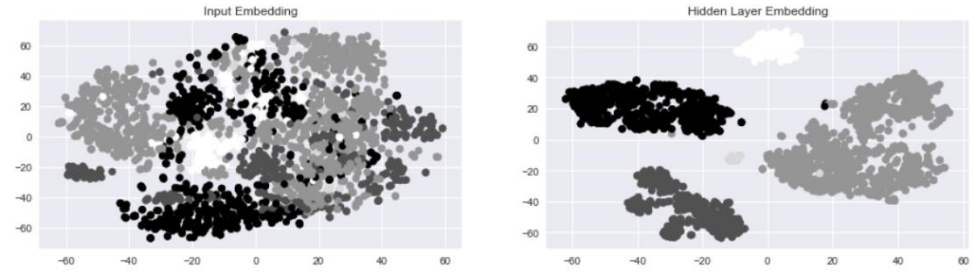


Figure 9training TNSE of NN

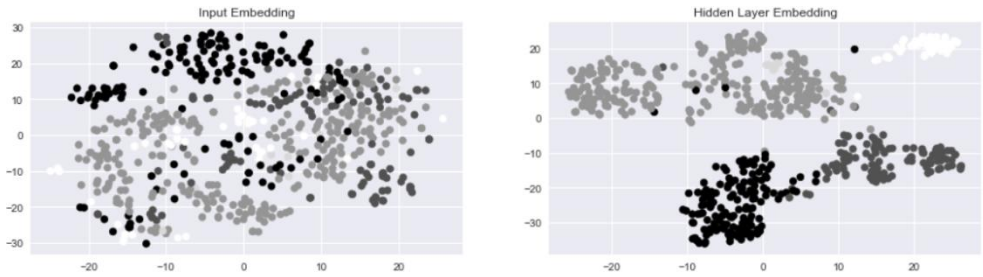


Figure 10 validation TNSE of NN