Traffic Sign Detection for Autonomous Vehicles

Ujjval Kavathia, Nachiketa Pathak, Sivanaga Surya Vamsi Popuri

Abstract: In autonomous driving, accurate detection of traffic signs is a crucial part. Accurate detection of traffic light can significantly reduce problems such as accidents on roads involving autonomous vehicles. Many algorithms and libraries have been developed to implement reliable and safe traffic sign detection. In this project we aim to review the performance of two state of the art neural network models namely Convolution Neural Networks and Recurrent Neural Networks and choose which is the best fit in terms of robustness and effectiveness for this particular application. This report presents the simulations done and the results obtained.

Keywords: Traffic Sign Detection, Neural Networks, autonomous driving

1 INTRODUCTION

Self-driving cars are the future of the automobile industry [1]. The intention of designing these technological marvels is to incorporate efficient, driver-less, and crash-proof vehicles. Since these vehicles are to be used in urban environments, with loads of traffic, they need to be able to detect traffic signals and signs on the road, to avoid accidents. Many state-of-the-art algorithms based on neural networks have been developed to address this problem. In this project, two such algorithms are identified, namely the Convolutional Neural Network [6] (CNN) based approach and the Recurrent-NN [5] based approach for traffic sign detection. Both algorithms trained using the images of traffic signs, and are tested on an input image to give an output, which is a prediction as to what the input image is. However, rather than just prediction, the performances of both methods are compared and analyzed in this project.

The rest of this report is organized as follows. Section 2 describes the methodology and theory, i.e. the procedures that have been carried out as well as the theory behind both types of neural networks. Section 3 contains the implementation of the project, including how the dataset was procured and the software used for coding. Section 4 contains the result section, which illustrates the results, which is a comparative study as to the performance of both methods. Section 5 concludes the report, with some information as to the possible future work that can be carried out pertaining to this research, and section 6 contains the references which are the supporting factors of this project.

2 METHODOLOGY

Up until now, the generated problem statement has been backed by relevant reference papers, journals, etc. In addition to these, open-source codes to extract datasets containing images of traffic signs of different shapes were taken [2]. These images are used to train the two types of neural networks. The reason these two learning models were chosen is due to the popularity of these algorithms amongst many researchers.

2.1 The Convolutional Neural Network

The hidden layers in the CNN consists of convolutional layers, pooling layers and normalization layers. In this type of neural network, convolutional and pooling layers are used as an activation functions. Convolution operates on two input signal such as two images, one of which acts as a filter. By convolution of these two images, the output can be obtained. The CNN can handle RGB images and can process them i.e. it has more feature compatibility. The following figure is a representation of a typical CNN.

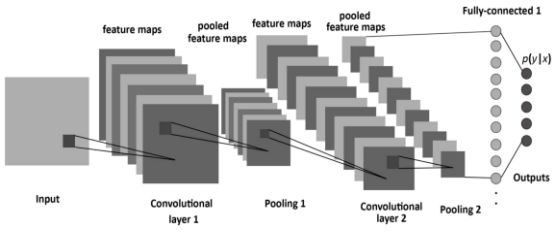

Figure 1: A typical CNN (Source: https://towardsdatascience.com/understanding-neural-networks-from-neuron-to-rnn-cnn-and-deep-learning-cd88e90e0a90)

In autonomous driving application, CNN can be used to detect road signs, traffic signals, lane detection, and other such specific objects on the road.

2.2 The Recurrent Neural Network

The Recurrent Neural Networks is recurrent as every layer inside RNN perform the same task for every input and hence, the output of each stage might depend on various previous inputs signals. Hence, RNN can be thought of a neural network with memory [5].

RNN, unlike the traditional feedforward network, utilize their internal memory to process input sequences that are arbitrary in nature. Compared to CNN, the RNN uses less features i.e. it has less feature compatibility. The following image is the structure of a typical RNN.

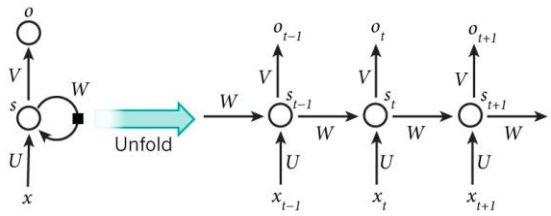

Figure 2: A typical RNN (Source: https://towardsdatascience.com/understanding-neural-networks-from-neuron-to-rnn-cnn-and-deep-learning-cd88e90e0a90)

In autonomous driving application, RNN can be used in the control elements where feedback is required such as automatic steering of vehicle. It can also be used to improve feedback and awareness of the driver by comparing data from various sources and taking the previous actions of driver into consideration.

2.3 The Problem Formulation

A dataset containing images of various traffic signs are obtained. This dataset is given to each neural network. Training is performed until both networks can recognize a given sign input to a reasonable extent.

The main goal is to observe the behavior of each Neural network when subject to change in three crucial network parameters, namely the. number of neurons in each layer, the type of activation function used, and the dropout percentage for each layer.

The changes in Accuracy, F1 Score, and Complexity for each NN is observed when the aforementioned parameters are tweaked, and hence an observation about the performance of each Neural Network is made.

3 IMPLEMENTATION AND SIMULATION

Since both the neural networks have to be trained first, a dataset is needed. Here, the dataset is extracted from German Traffic Sign Recognition Benchmark, which is available for open source download from Kaggle [2].

The dataset contains 43 different classes of road signs, such as speed limits, stop signs, etc. There is a total of 39,209 images in the training dataset distributed among the 43 road signs, which means about 906 images for each type of sign. There are 12630 images in the testing dataset.

Each image has been resized to a size of 30x30 to minimize computational complexity as much as possible. For each image the pixel values are scaled by dividing by 255 to get better normalization. The training data is divided into a split of 80% for training set and 20% for validation set. The following image shows the 43 different traffic signs.

One can note that for a particular class of road sign, different types of images have been collected and stored in the dataset. For example, for the speed limit sign of 30mph, we can see clear as well as distorted images in the given dataset.

Figure 3: The different types of road signs such as direction changes, speed limits, stop signs, etc.

Coming to the software to be used, both neural networks, the CNN and RNN can be implemented in python using Tensorflow, and its sub functions. We initially assign for both NNs 3 hidden layers, ReLU activation function, and a drop out rate of 0.25. The number of epochs is set to 10.

We first change the number of neurons in both networks, keeping the activation function and drop out rate constant. Then we repeat the same, by changing the drop out rate for both NNs, keeping the other two parameters constant, and finally deal with changing the activation function for both NNs. The impact of these changes on the accuracy, F1 Score and Complexity are recorded for each execution of the program.

For each parameter change, the code is run 4 times on python for each neural network, and the average Accuracy, F1 score and complexity are computed. The code is run multiple times in order to verify the authenticity of the results obtained, and to make sure that the results obtained are certain and don't change much.

4 RESULTS

The following section provides figures and tabular columns which were obtained after training was done on both neural networks for different configurations. The results are as follows.

4.1 Initial Results

This pertains to the performance of both methods, when the network parameters were set to the default values as mentioned by the tensorflow python documentation []. In other words, the configuration of the parameters were set as follows.

Parameter	Value
# Neurons in H1	32
# Neurons in H2	64
# Neurons in H3	64
Drop-out rate	0.25
Activation Function	ReLU

Table 1: The initial parameters set for both NNs by default.

Figure 4 shows the accuracy and loss for training and validation, for the CNN.

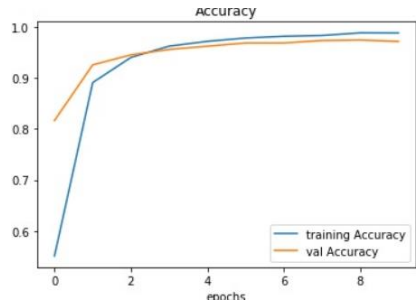

Figure 4 (a): Accuracy for CNN

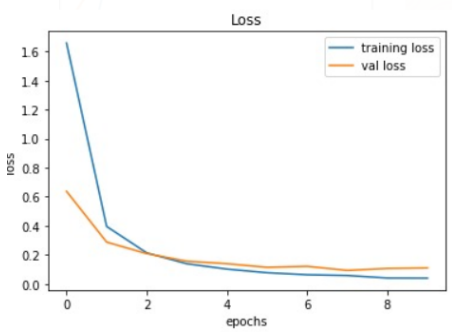

Figure 4 (b): Loss for CNN

And now, the same was performed for RNN and the results obtained were as follows in Figure 5.

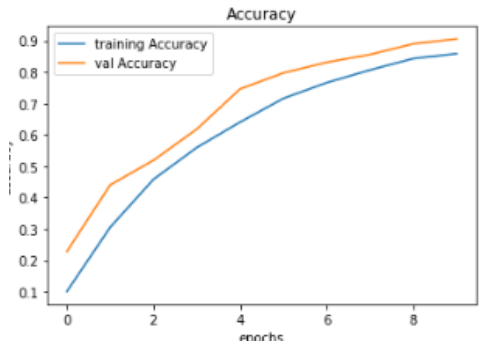

Figure 5 (a): Accuracy for RNN

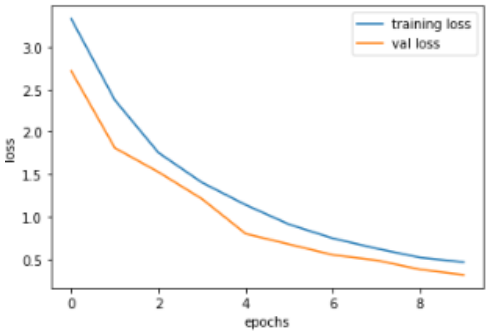

Figure 5 (b): Loss for RNN

And now, the accuracy, F1-Score and the Complexity (Average Values) were recorded and are obtained as follows.

Measure	CNN	RNN
Accuracy	93.82%	74.53%
F1-score	90.74%	63.75%
Complexity	190.3 sec	273.25 sec

Table 2: Performance of the NNs

4.2 Increase in number of neurons

Now, the number of neurons in the hidden layers were increased to the following values shown in table 2, keeping the drop-out rate and the activation function constant, for both the neural networks.

Parameter	Value
# Neurons in H1	128
# Neurons in H2	128
# Neurons in H3	64
Drop-out rate	0.25
Activation Function	ReLU

Table 3: The initial parameters set for both NNs by default.

And as a result, the following changes were noted in the performance of both neural networks.

Measure	CNN	RNN
Accuracy	92.4%	*84.53%*
F1-score	90.24%	*75.75%*
Complexity	*864.5 sec*	*573.25 sec*

Table 4: Performance of the NNs

The changes made here have caused the CNN's complexity to increase, while it's performance is more or less the same. On the other hand**, the accuracy of RNN increased significantly with respect to the default.**

4.3 Decrease in Drop-out rate

The drop out rate was reduced to 0.15, keeping all others fixed, as shown in table 5.

Parameter	Value
# Neurons in H1	32
# Neurons in H2	64
# Neurons in H3	64
Drop-out rate	0.15
Activation Function	ReLU

Table 5: The initial parameters set for both NNs by default.

For the changes made here, the following results were obtained.

Measure	CNN	RNN
Accuracy	94.4%	*78.75%*
F1-score	91.5%	*70.75%*
Complexity	280.73 sec	*283.45 sec*

Table 6: Performance of the NNs

The changes made here have caused the CNN's performance to be more or less the same. On the other hand, the **Accuracy of RNN has increased slightly** in comparison to the default, and so has its complexity.

4.4 Increase in Drop out rate

The drop out rate was increased to 0.4, keeping all others fixed, as shown in table 5

Parameter	Value
# Neurons in H1	32
# Neurons in H2	32
# Neurons in H3	64
Drop-out rate	0.4
Activation Function	ReLU

Table 7: The initial parameters set for both NNs by default.

And as a result, the following changes were noted in the performance of both neural networks.

Measure	CNN	RNN
Accuracy	90.4%	*62.75%*
F1-score	88.5%	*44.75%*
Complexity	264.3 sec	285.7

Table 8: Performance of the NNs

The changes made here have caused the CNN's performance is more or less the same. On the other hand, the **accuracy of RNN decreased significantly** with respect to the default, but complexity is around the same value.

4.5 Change in activation function – 1

The activation function was changed to sigmoid from RelU, keeping all other parameters fixed, and the following results were obtained.

Measure	CNN	RNN
Accuracy	82.4%	*24.45%*
F1-score	74.5%	*9.5%*
Complexity	266.39 sec	297.88

Table 9: Performance of the NNs

Changing the activation function to sigmoid caused the RNN to give its worst performance ever. The performance of CNN also became poor, but only slightly poor.

4.6 Change in activation function – 2

The activation function was changed to "TanH" from RelU, keeping all other parameters fixed, and the following results were obtained.

Measure	**CNN**	**RNN**
Accuracy	95.7%	*85.4%*
F1-score	93.5%	*79.5%*
Complexity	275.38 sec	*300*

Table 10: Performance of the NNs

Changing the activation function to TanH caused the RNN to give its best performance ever, both in terms of accuracy scores and complexitt. The performance of CNN has been improved by this operation.

From this, certain interesting observations can be deducted. From the results obtained for both the implementations, the general trend is that the *CNN turns out to be more robust, when it comes to changing the network parameters i.e. its performance changes only slightly.*

On the other hand, the *RNN turns out to be more sensitive, as there have been situations when the accuracy was extremely poor, as well as relatively high with lesser complexity, for small changes in parameters.*

However, this by any way does not mean that the RNN is not a good choice. This just means that one needs to be extremely careful when designing an RNN for their application, as small changes in the network parameters can cause huge differences in the desired outputs.

5 CONCLUSIONS

Thus, as concluded in this report, the fundamental step of detecting the traffic signs has been achieved with two neural networks. Dataset containing the signs were fed to both networks and trained. The effect of changing network parameters on the performance of both networks were recorded.

An inference as to how each neural network behaves with respect to each other was made, which tells us about how both methods behave with respect to each other under different conditions and circumstances.

In the future, now that we have noted the behavior of the RNN with respect to CNN, we would like to extend our research by trying out RNN on video streams taken from the real-world containing traffic signs, and tuning the neural network parameters in favor of lesser complexity and more accuracy.

Also, we aspire to observe more trends in the performances of both neural networks, by exploring more network parameters to tweak.

6 REFERENCES

[1] M. V. Rajasekhar and A. K. Jaswal, "Autonomous vehicles: The future of automobiles," 2015 IEEE International Transportation Electrification Conference (ITEC), Chennai, 2015, pp. 1-6.

[2]https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign, Traffic sign dataset

[3]https://pythonprogramming.net/introduction-deep-learning-python-tensorflow-keras/, Tutorial for Python based deep learning

[4]https://towardsdatascience.com/recognizing-traffic-signs-with-over-98-accuracy-using-deep-learning-86737aedc2ab, Python for Deep learning.

[5] A. N. Michel, "Recurrent neural networks: overview and perspectives," Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS '03., Bangkok, 2003, pp. III-III.

[6] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), Antalya, 2017, pp. 1-6.

[7]https://towardsdatascience.com/understanding-neural-networks-from-neuron-to-rnn-cnn-and-deep-learning-cd88e90e0a90