Predicting car movement for autonomous driving through traffic using neural networks

Camilio Delfgaauw
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
c.delfgaauw@student.utwente.nl

ABSTRACT

The technology of autonomous driving is becoming a more common technology. These autonomous cars have the ability to communicate with other autonomous cars. This makes it possible to use the data of the autonomous cars to make predictions on the traffic around the cars. Because of this this research tries to determine what deep learning method can be used in the prediction of vehicle movement. And from the method that can make predictions which method works best. The goal of the research is to answer the following research questions: Q1 Which deep learning method is best to use in predicting traffic movement for autonomous driving? Q2 What machine learning techniques can be used in the trajectory and speed prediction of traffic? Q3 Which method creates the smallest error in our simulated setup? To answer these question the literature study determined to use a CNN method where you feed the network spatial data through the use of images and an LSTM network that uses coordinate data from each vehicle individually to determine the next coordinate of the vehicle. The generate data than can be used in the training and testing of these deep learning techniques, SUMO was used. With the help of SUMO two data sets where created, one with a simplistic network to test the different techniques and one more realistic and complicated to determine which technique preforms best. The CNN was moderately accurate with an accuracy of around 77% after training for 150 epochs. But the CNN also produced a lot of false positives making the results diluted. The LSTM network preformed slightly worse with an average accuracy of 58%. But these results where not influenced by any false positive values, because of how the network works. The above mentioned values together with some more variables, the conclusion was made that the LSTM network is more suitable for the predicting of traffic movement for autonomous driving.

Keywords

Autonomous driving, deep learning, SUMO, CNN, LSTM

1. INTRODUCTION

Tesla is one of the first companies that have introduced an auto pilot in there manufactured cars. Next to Tesla there are multiple companies that have vehicle being driven by AI. These Autonomous vehicle are becoming better. But they also raise question on how they could improve. One of the things that could be improved on a autonomous vehicle is it's awareness of it's surrounding. If there are multiple autonomous cars in a road network, they could communicate their positions to help other autonomous cars in their navigation. Having the position of other vehicles around the autonomous vehicle is not good enough, because you want to determine where the cars are going to and if this influences your path or route. For this purpose this research tries to answer the following research questions: Q1 Which deep learning method is best to use in predicting traffic movement for autonomous driving? Q2 What machine learning techniques can be used in the trajectory and speed prediction of traffic? Q3 Which method creates the smallest error in our simulated setup? In the first part of the research we have preform a literature study. In this study we want to determine what previous research has been preformed on this subject and what deep learning method are best candidate for the goal of this research. The next part of the research was data collection. For the collection several networks where created in sumo. SUMO is a mobility simulator that can generate vehicle movement in a road network. These data sets are being used in two deep learning methods, the first is a convoluted neural network and the second is a Long Short term Network. These networks have been created in Python using Tensorflow and Keras. The end of this paper will discus the results and make conclusions based on the results

28th Twente Student Conference on IT July. 3th, 2020, Enschede, The Netherlands.

2. LITERATURE STUDY

The literature study tries to answer the research question Q2. The literature study identified several articles that preform predictions on traffic flows [4] [5][6]. Most of these article use liniar regression to predict traffic flows. There are several articles that use deep learning methods in the prediction of traffic flow. One of the most interesting article is the article by MA 2017.[6]. This article describes at technique to evaluate traffic flow as images. These images could then be used to predict the next state of the network using a Convolutional neural network (CNN). The article of Bratsas 2019 [3] preformed a comparison on different machine learning methods for traffic speed prediction. In his article Bratsas states that the Multilayer Perceptron model(MLP) works best for circumstances with great variations. Because a CNN works faster and is more accurate than a MLP this research focuses on the use of a CNN. With the use of images to represent the network over time. Another interesting article is the article by Chen 2017 [5]. In this article they propose a LSTM network to predict short term traffic flow in a network. LSTM are very good in predicting time series and therefore are interesting in

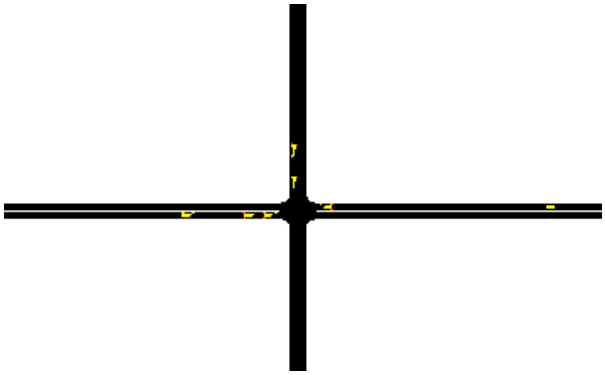

Figure 1. hand drawn road network in SUMO, showing the intersection of the 4 roads

Figure 2. network generated using Netconvert from data of OpenStreetMap of the University of Twente campus

the use of vehicle predictions on an individual level. This makes the LSTM one of the methods used in this research. Next to this articles where identified on the subject of collision detection [1] [2] [7], these articles take a same approach as this research towards predicting speed and direction of specific vehicles. The article do not use deep learning techniques and are therefore different from the goal of this research. Based on the literature study a CNN and a LSTM network where selected as the best methods to use for the prediction of traffic flows. The selection of these two networks show to very different approaches making it interesting to compare them to determine the best deep learning method.

3. SUMO

3.1 Network generation

This reseach needs a data set that represents a road network and a data set that represents traffic on this road network. To generate this data SUMO was used. SUMO is a Simulation Of Urban Mobility, crated by Institute of Transportation Systems at the German Aerospace Center. It is an open source software. Because it is an open source software it has a large user basis making it easy to find tutorials, help and tips. Within SUMO there is the option to generate networks. These networks can be generated by drawing them by hand or they can be generated using Netconvert. This a tool allows the user to generate a network based on data collected from OpenStreetMap. Netconvert makes it possible to take any place from around the world and convert it in to a network within SUMO. For the purpose of this research two networks where created. On drawn by hand (See Figure 1) and one generate with Netconvert using a OpenStreetMap (See Figure 2). The first network has been chosen because it represents a very simplistic network with 4 roads and one intersection. The intersection is managed by a traffic light. This very simplistic network allows us to generate a base line for the deep learning methods that will be implemented. To test the durability of the deep learning methods the second data set was chosen to be more complicated, and represent a real life scenario. Because of the familiarity the campus of the University of Twente was choosen as the second road network.

3.2 Traffic generation

The created networks can be combined with a set of routes that are used to generate the individual vehicles. These routes can be generated automatically with the RandomTrips.py script. For the first example this script was used to generate random vehicle movement within the network. For the second example the routes were first generated by RondomTrips script, after the creation known routes where added. Because the area is known to the researchers improvement could be made by adding routes that are more commonly used in this area. This made the example less random and more realistic.

3.3 Output generation and post processing

By combining the networks and the routes, SUMO can generate the traffic by simulating the vehicles on the roads in the network. SUMO does this by making a snapshot of the network status every second. This allows SUMO to generate an XML file with the position and speed of every vehicle in the network on every time-step. From these XML files the location of every vehicle at every time-step could be extracted.

4. DEEP LEARNING METHODS

Based on the literature study two deep learning methods where implemented for this research. The first method is convoluted neural network that takes images from the whole network for every time-step in the example. The second method is long short term memory network, that uses the locations data of the vehicles as times series data. This makes it two very distinct approaches to the problem. In the first approach the method looks at the network as a whole making it possible to learn from the flow of traffic, where as in the second approach the method looks at the vehicle level and determine the next position based on the previous positions.

4.1 Convolutional Neural Networks

For the CNN network Tensor was used to generated and train the network. The pix2pix example was used as a guidline in the creation of the neural network. The network uses 2 dimensional pictures as both input and output. With the use of Python pictures where generated for every time-step in our examples. These pictures where generated in 2 types. The first type used a black background with colored pixel for every vehicle in the network. For the second type we used the network as a background in red after which the cars where added in blue. This works well because an image has 3 dimensions for each pixel (red, green and blue), this makes it possible for the network to differentiate between the roads and the vehicles. To make the data set the pictures were linked based on their time-step. For the purpose of training and testing the network these pictures where generated in doubles by combining the picture of time-step n with the picture of time-step n+1. This allowed the network to generate a

Table 1. First set of layers of CNN showing the structure of the generator

Layer type	input shape	output shape
Input layer	(1024, 1024, 3)	(1024, 1024, 3)
LeakyReLU	(512, 512, 64)	(512, 512, 64)
Conv2D	(512, 512, 64)	(256, 256, 128)
BatchNormalization	(256, 256, 128)	(256, 256, 128)
LeakyReLU	(256, 256, 128)	(256, 256, 128)
Conv2D	(256, 256, 128)	(128, 128, 256)
BatchNormalization	(128, 128, 256)	(128, 128, 256)
LeakyReLU	(128, 128, 256)	(128, 128, 256)
Conv2D	(128, 128, 256)	(64, 64, 512)
BatchNormalization	(64, 64, 512)	(64, 64, 512)
LeakyReLU	(64, 64, 512)	(64, 64, 512)
Conv2D	(64, 64, 512)	(32, 32, 512)
BatchNormalization	(32, 32, 512)	(32, 32, 512)
LeakyReLU	(32, 32, 512)	(32, 32, 512)
Conv2D	(32, 32, 512)	(16, 16, 512)

picture based on time-step n (the input) and than correct the generator to create the Output of time-step n+1 (the target). The generator is build of several layers that are being used to extract features to make the connections between the input and the target (See Table 1). These layers consist of 3 types of layers: Convolutional layers, ReakyRelu layers and Batch normalization. The convolutional layers make the connections between the input and the output to make the generator trainable. ReakyRelu layers are used to make the values that are 0 slightly bigger so the network can make connections between the data points with 0 values. The Batchnormalization layers make all the values between 0 and 1, this is needed for the neural networks to make the calculation on the connections in the network. The layers follow each other until the shape of the last layer is (1,1, 512) this makes it possible for each pixel to be connected in the data set to the generated output. After this layer the data will be concatenated using the output of each of the convolutional layers to generate the full size of the full image again. This last layer can be used to verify the accuracy with the target image of the data set.

For every variation of the input the generator trained for 150 epoch on a data set size of 180 pictures for the first example and 439 pictures for the second example. The other part of the data sets where being used to test the generator on its accuracy.

4.2 Long Short Term Memory network

The LSTM network is build in Keras. This network is trained and test with csv files containing every position of each vehicle in every time-step. These data sets where divided between a training set and a test set. The model was build with a hidden layer of 16 LSTM blocks. The network was trained for 150 epochs to match the CNN. Because the data consists of coordinates, the model runs on 2 dimensional data representing the x and the y coordinate of the vehicle.

5. RESULTS

5.1 CNN

The generator of the CNN network was trained and test on 4 different data sets. The first data set was the simplistic network with only one intersection. The second data set was the simplistic network containing a layer with the network itself. The third data set was the realistic data

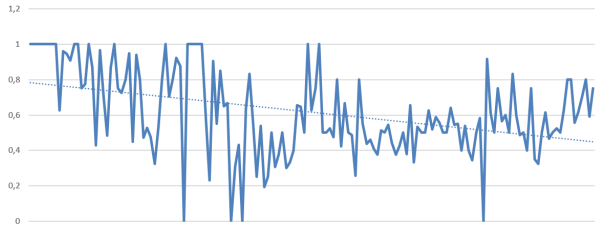

Figure 3. Accuracy based on the amount of predicted points divided by the amount of ground truth points during the training phase

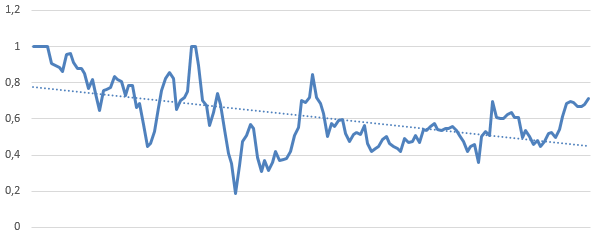

Figure 4. Moving average of the accuracy based on the amount of predicted points divided by the amount of ground truth points during the training phase

set which was base on the campus of the university. The last data set was the realistic data set with the network layer as a dimension in the data.

5.1.1 Simplistic network example

During the fitting of the CNN network to the training data, the accuracy of the model was test on a random example from the testing data to determine the accuracy during the training's phase. From these random examples the amount of points in that represent vehicles in the image was calculated, the amount of predicted points and the amount of points that are positive in the ground truth image and the prediction image was calculated. This data determines the accuracy of the generator by dividing the amount of good predicted points by the amount of points that are positive in the ground truth image. Plotting these values generate a picture showing the accuracy during the training's phase. Figure 3 shows that the accuracy. This graph is very volatile and hard to read. A moving average was made of the data to make it easier to read and understand. Figure 4 shows the moving average of the accuracy. The accuracy is steadily declining. This decline can be explained by the declining amount of generated points. This happens because the network is trying to adjust it's output to the ground truth. Figure 5 shows the amount of generated points divided by the amount of positive ground truth points, these points are called false positives. The amount of false positives is steadily increasing, this means that the generated model has a problem with generating more bad points.

There is a method to decrease the amount of bad generated points. By implementing a threshold, the generate image can ignore points that have a lower value. The ground truth image has a pixel value of 255 for it's blue dimension, this means that if the generated point also has a value of 255 for it's blue dimension this is a perfect match. By implementing the threshold 127,5 for it's blue dimension all the points that have been decreased in intensity by

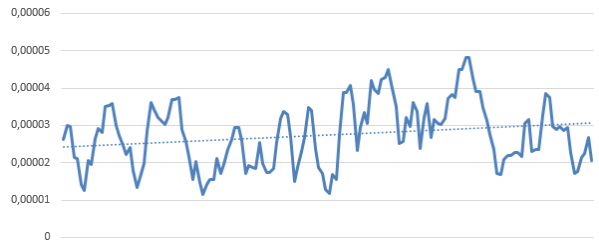

Figure 5. Moving average of the amount of positive points in the ground truth per predicted points during the training phase

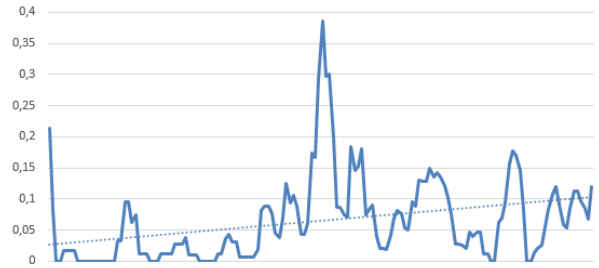

Figure 6. Moving average of the accuracy based on the amount of good predicted points divided by the amount of ground truth points with the implemented threshold during the training phase

the generator can be eliminate. During the learning process the network lowers the value of bad predicted points making it possible to eliminate them with the threshold. Figure 6 and 7 show that, with the implemented threshold the accuracy of the model is increasing and the amount of false positives is decreasing.This shows that the prediction is increasing during the training phase.

After fitting the CNN network on the training data, the generator was tested on the test data. The confusion matrix shown in table 2 gives the result of the testing data set. This gave an average of 0,66 good generated point for every ground truth point. The generator also generated a lot of false positives similar to the training phase, the amount of false positive points was on average 93.4205 per image. This means that the generated image are still distorted with random point or points around the vehicle after the training phase. By implementing the threshold on this data as well, the amount of random points where decreased to an average of 25 but the accuracy to dropped

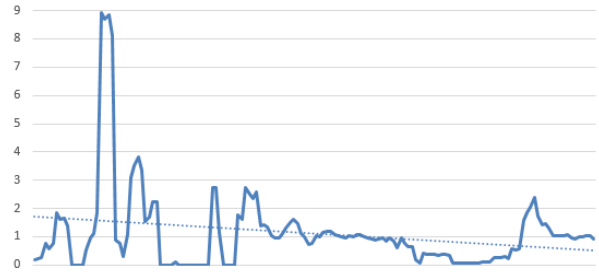

Figure 7. Moving average of the amount of positive points in the ground truth per predicted points with the implemented threshold during the training phase

Table 2. Confusion matrix of the simplistic example

	without threshold		with threshold	
	predicted positive	predicted negative	pred. positive	pred. negative
Ground truth positive	14.34	7.27	0.84	20.77
Ground truth negative	934,205	114,349	25.54	1,048,529

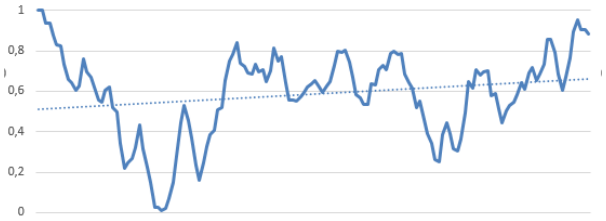

Figure 8. Moving average of the accuracy based on the amount of predicted points divided by the amount of positive points in the ground truth with the implemented threshold for the training set with road layer during the training phase

to 0.04 good generated point for every positive point in the ground truth.

To improve the predictions of the neural network an extra layer has been implemented in the images. This layer represents the roads of the network in the red dimension. The idea of this is that by giving the generator data about the model this would increase the accuracy.

Figure 8 and 9 show the accuracy and the amount of false positives per positive ground truth point. These figures show that the during the training's phase the accuracy is increasing and that the rate of false positives is decreasing. this shows that the generator is preforming better on the data that has the network implemented as a layer than on the data without this. To make the data comparable the threshold was also implemented on the data with the network. Figure 10 and 11 show the accuracy and the ratio of false positives with the implemented threshold. With the threshold the accuracy is still increasing but the rate of false positives does not seem to decrease, this means that with more training there is a chance that this generator still produce a lot of random points.

After fitting the CNN network on the training data with the extra layer, the generator was tested on the test data.

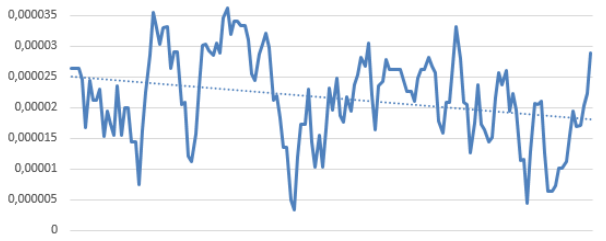

Figure 9. Moving average of the amount of positive ground truth points per predicted points with the implemented threshold for the training set with road layer during the training phase

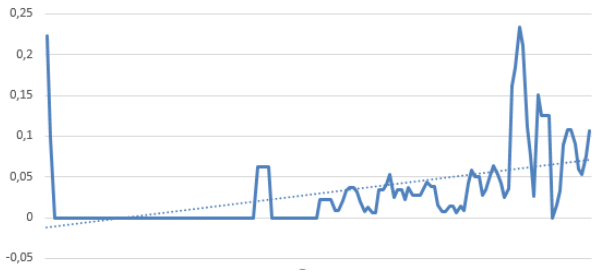

Figure 10. Moving average of the accuracy based on the amount of predicted points divided by the amount of positive ground truth points with the implemented threshold for the training set with road layer during the training phase

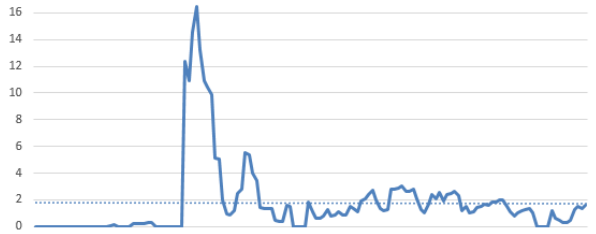

Figure 11. Moving average of the amount of positive ground truth points per predicted points with the implemented threshold for the training set with road layer during the training phase

Table 3. Confusion matrix of the simplistic example with road network layer

	without threshold		with threshold	
	predicted positive	predicted negative	pred. positive	pred. negative
Ground truth positive	17.36	4.61	2.34	19.63
Ground truth negative	1,045,062	3,488	55	1,048,499

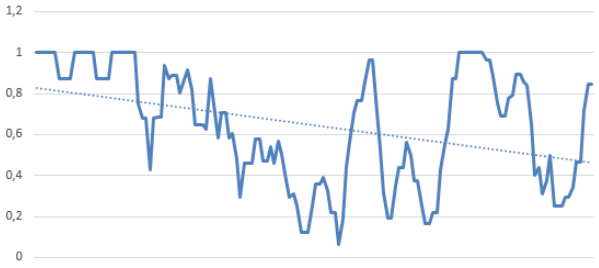

Figure 12. Moving average of the accuracy based on the amount of predicted points divided by the amount of positive ground truth points with the more realistic data set during the training phase

The confusion matrix shown in table 3 gives the result of the testing data set. This gave an average of 0,77 good generated point for every positive ground truth point. The generator also generated a lot of false positives similar to the training phase, the amount of false positive points was on average 1.045.065 per image. This means that these generated image were more distorted with random point or points around the vehicle after the training phase than the example without the network layer. By implementing the threshold on this data as well, the amount of random points where decreased to an average of 55 but this also dropped the accuracy to 0.11 good generated point for every positive ground truth point. This is significantly higher than the example without the network layer.

5.1.2 Realistic network example

There where several issues with running this data set. The original idea was to run this data set with images of 4000 by 4000 pixels because the network was much larger and contained interesting movement. However the computational power of the devices at hand where not sufficient enough to create a CNN for images this large. Even with the use of Google COLAB the examples where to be to be calculated in meaning full time. Because of this the images had to be decreased to a size of 1024 by 1024 pixels which is similar to the size of the simplistic network. Because the more realistic data set had to be down scaled the amount of vehicles in the network also decreased. This meant that the average amount of vehicles was only 4 per image. This made the generator a lot less accurate.

Figure 12 and 13 show the accuracy and the amount of false positives per positive ground truth point during the training's phase. The images show that with increased training the accuracy decreases but also the ratio of false positives. To compare this to the results of the other data sets the threshold was implemented. Figure 14 and 15 show the accuracy and the false positives ratio with the threshold implemented. In the last two figures the low amount of vehicles per image is becoming a problem. The generator has to few vehicles to make good predictions and because of this the accuracy drops to almost 0.

After fitting the CNN network on the training data, the generator was tested on the test data.The confusion matrix shown in table 4 gives the result of the testing set. This gave an average of 0,45 good generated point for every positive ground truth point. The generator also generated a lot of false positives similar to the training phase, the amount of false positive points was on average 1.046.526 per image.This is very similar to the simplistic data set. By implementing the threshold on this data as well, the amount of random points where decreased to an

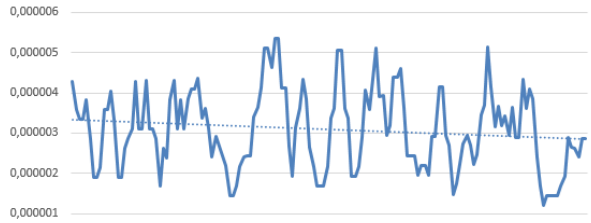

Figure 13. Moving average of the amount of positive ground truth points per predicted points with the more realistic data set during the training phase

Figure 14. Moving average of the accuracy based on the amount of predicted points divided by the amount of positive ground truth points with the more realistic data set and the threshold during the training phase

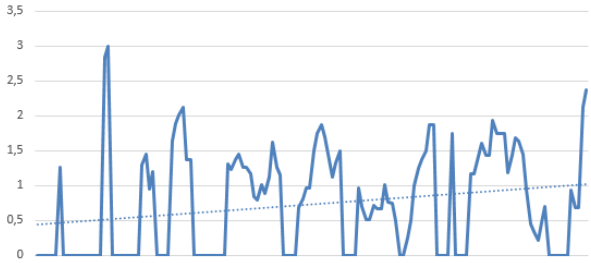

Figure 15. Moving average of the amount of positive ground truth points per predicted points with the implemented threshold for the training set with the more realistic data set and the threshold during the training phase

Table 4. Confusion matrix of the realistic example

	without threshold		with threshold	
	predicted positive	predicted negative	pred. positive	pred. negative
Ground truth positive	1.57	1.77	0	3.34
Ground truth negative	1,046,526	2,046	3	1,048,570

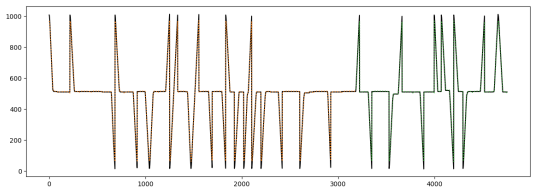

Figure 16. The X coordinates of the ground truth for the simplistic example (grey), training data set (orange) and the testing data set (green)

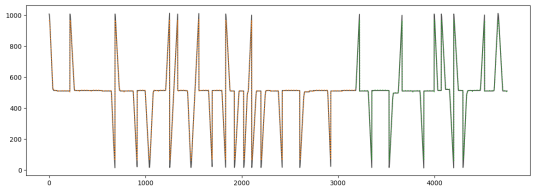

Figure 17. The Y coordinates of the ground truth for the simplistic example (grey), training data set (orange) and the testing data set (green)

average of 3, this looks really promising. But the accuracy was decreased to 0, because there where no points with a value above the threshold. This makes the results with the threshold not comparable to the simplistic data set. More testing is needed to determine the effectiveness of the CNN with a realistic example.

5.2 LSTM

The LSTM network produces a mean square error for the difference between the generated coordinates and the ground truth coordinates. For the very simplistic network the MSE was 45.13 for the training set and 48.02 for the testing set. Next to this the average distance between the generated locations and the vehicle locations was calculated to be 18.27 meters. To determine the amount of correct predictions we compared the generated and the ground truth and determined the amount of positions within 1 meter. This gave an accuracy of 0.59 for each position. Figure 16 and 17 where generated using respectively the x and y coordinates for the original data set in black the predicted training data set in orange and the predicted testing data set in green. These images show that the fitting to the original data set is good, but at the peaks of the graph's the lines separate. This means that the extreme values are harder for the LSTM to predict.

For the more realistic network the MSE was 78.21 for the training set and 79.93 for the testing set. This means that the accuracy is decreased within the more complex network. Next to this the average distance between the generated points and the vehicle coordinates was calculated to be 27,91 meters. And if we look at the ratio of correct predicted locations within 1 meter the ratio is 0.18. This is a significant drop in accuracy compared to the simplistic network. This means that the network needs to be trained longer for the more complicated road network. Figure 18 and 19 where again generated for the x and y values. These graph's look much more fitting, but this is caused by the higher range of the data points that are being used. Ass the numbers show the fitting is less accurate than the simplistic network.

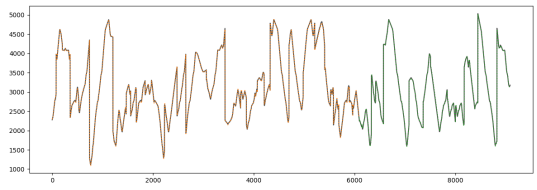

Figure 18. The values of the X coordinates of the realistic example from the original data set (grey), training data set (orange) and the testing data set (green)

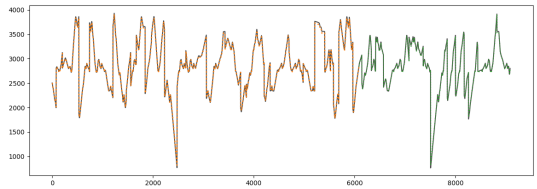

Figure 19. The values of the Y coordinates of the realistic example from the original data set (grey), training data set (orange) and the testing data set (green)

6. CONCLUSION

Table 5 and 6 shows the accuracy results for all experiments. The training results generated by the CNN show that the prediction of vehicle movement in a road network is possible. Because of the decrease in false positives and the increase in accuracy during the training phase, the expectations are that the network will preform better after more training is done. The data also shows that there is a lot of random points being generated by this method that needs to be removed. This has been achieved by implementing a threshold, which removes data with lower point values. Next to this the CNN produced less promising result using the more realistic data set. The origin of this problem lies in the size of the road network, because the size was to big. This resulted in smaller part of the network being used in the training and testing of the model. This caused the number of vehicles in each time step to be very low. this made it harder for the model to train and harder to test the result.

The results generated by the LSTM show that the model is a good candidate for the prediction of vehicle movement in a road network. The model could generate a error distance of 18 meters and 28 meters respectively for the simplistic data set and the more realistic data set. By looking at the amount of correct identified locations within 1 meter of the ground truth this is 59% and 18% respectively.

Table 5. Results of the CNN network

	CNN		CNN with threshold	
	simplist example	realistic example	simplistic example	realistic example
without road network	66%	45%	4%	0%
with road network	77%		11%	

Table 6. Results of the LSTM network

	LSTM	
	simplist example	realistic example
without road network	59%	18%
average distance to vehicle position (m)	18	28

The two methods can be compared on several factors:

The training duration. The training duration for the CNN model is a lot longer than for the LSTM model. The training duration of the CNN is on average between 7 and 9 hours where the training duration of the LSTM is between 30 minutes to an hour. This makes the LSTM model much easier to use.

The accuracy of good predicted locations. The accuracy of the CNN was between 45% and 77%. The higher amount of accuracy was achieved by having a lot of false positives. after removing the false positives by using a threshold the accuracy was between 0% and 11%. The accuracy of the LSTM was 59% and 18%. Because the LSTM model produces no false positives this accuracy is not influenced by this. This answers research question Q3.

By taking these factors in consideration the LSTM has been selected as the best model to use in the prediction of vehicle movement in a network. This answers the primary research question of this research.

7. FUTURE RESEARCH

The results of this research show that the LSTM model preforms best for this research goal. But this conclusion could be reinforced more by preforming some more calculations. Because all the test where trained with 150 epochs we do not know how much training is needed to generate a model that preforms above 90% and has no, to almost no false positives. Next to this the models could be improved by using other variables such as the speed of the vehicles and the past location of each vehicle. Also the implementation of the model in real life scenarios could be test better by having more realistic model that could be generated using real life traffic data to make the model a better fit for the real world.

8. REFERENCES

[1] D. Althoff, D. Wollherr, and M. Buss. Safety assessment of trajectories for navigation in uncertain and dynamic environments. In *2011 IEEE International Conference on Robotics and Automation*, pages 5407–5412, May 2011.

[2] G. Aoude and J. How. Using support vector machines and bayesian filtering for classifying agent intentions at road intersections. 09 2009.

[3] C. Bratsas, K. Koupidis, J. M. Salanova Grau, K. Giannakopoulos, A. Kaloudis, and G. Aifadopoulou. A comparison of machine learning methods for the prediction of traffic speed in urban places. *Sustainability*, 12:142, 12 2019.

[4] S. Chavhan and P. Venkataram. Prediction based traffic management in a metropolitan area. *Journal of Traffic and Transportation Engineering (English Edition)*, 2019.

[5] W. Chen, Z. Zhao, J. Liu, P. C. Y. Chen, and X. Wu. Lstm network: A deep learning approach for

short-term traffic forecast. *IET Intelligent Transport Systems*, 11, 01 2017.

[6] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang. Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction. *Sensors*, 17:818, 04 2017.

[7] K. Okamoto, K. Berntorp, and S. D. Cairano]. Driver intention-based vehicle threat assessment using random forests and particle filtering. *IFAC-PapersOnLine*, 50(1):13860 – 13865, 2017. 20th IFAC World Congress.