Using Recurrent Neural Networks for Action and Intention Recognition of Car Drivers

Martin Torstensson[1], Boris Duran[2] and Cristofer Englund[2,3]
[1]*Chalmers University of Technology, Gothenburg, Sweden*
[2]*RISE Viktoria, Lindholmspiren 3A, SE 417 56 Gothenburg, Sweden*
[3]*Center for Applied Intelligent Systems Research (CAISR), Halmstad University, SE 301 18 Halmstad, Sweden*

Keywords: CNN, RNN, Optical Flow.

Abstract: Traffic situations leading up to accidents have been shown to be greatly affected by human errors. To reduce these errors, warning systems such as Driver Alert Control, Collision Warning and Lane Departure Warning have been introduced. However, there is still room for improvement, both regarding the timing of when a warning should be given as well as the time needed to detect a hazardous situation in advance. Two factors that affect when a warning should be given are the environment and the actions of the driver. This study proposes an artificial neural network-based approach consisting of a convolutional neural network and a recurrent neural network with long short-term memory to detect and predict different actions of a driver inside a vehicle. The network achieved an accuracy of 84% while predicting the actions of the driver in the next frame, and an accuracy of 58% 20 frames ahead with a sampling rate of approximately 30 frames per second.

1 INTRODUCTION

Traffic safety is today one of the major global societal challenges and traffic accidents has become one of the most common causes of death among young persons (World Health Organization (WHO), 2015). Human errors are one of the major factors affecting the situations leading up to traffic accidents (Singh, 2015). Among the systems used today to reduce human errors are warning systems. Examples of these warning systems are Driver Alert Control, Collision Warning and Lane Departure Warning[1]. The intention of such systems is either to alert the driver of a hazardous situation or about a certain condition. This study explores ways to further improve such systems by taking into account the future state of the driver.

1.1 Related Work

Recurrent neural networks (RNN) with long short-term memories (LSTM) have previously been used by (Jain et al., 2016), (Olabiyi et al., 2017) for driver action prediction, to predict for example breaking and lane changes. Methods based on sensor fusion were applied to data from e.g. the CAN bus, GPS and cameras. The camera input did, in both cases, contain at least one camera directed at the driver's face. In (Jain et al., 2016) a precision of 90.5% and a recall of 87.4% was achieved with 3.16 seconds-to-maneuver. The focus of these studies are maneuvers and even though images of the driver were used to predict these maneuvers they did not cover the state of the driver.

Another method used for prediction of lane changes was presented in (Pech et al., 2014). The main concept of the study was to use eye gaze of the driver for predictions ten seconds in advance. The information was, in turn, extracted from the angle and position of the driver's head. An overall accuracy of 75% was reached. This type of methodology provides a way to make predictions a long time in advance however it is sensitive to the position of the drivers head and what windows the driver looks through and what mirrors the driver use. In the case of actions inside the vehicle, the drivers focus on windows and mirrors might not provide the same information as the driver might be looking at the road while trying to reach something inside of the car.

Other studies have focused on the behavior of the driver, such as (Carmona et al., 2015) a study of the driver's level of aggressiveness in different environments. The inputs used were mainly based on the

[1]https://volvo.custhelp.com/app/manuals/ownersmanualinfo/year/2018/model/V60

Torstensson, M., Duran, B. and Englund, C.
Using Recurrent Neural Networks for Action and Intention Recognition of Car Drivers.
DOI: 10.5220/0007682502320242
In *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2019)*, pages 232-242
ISBN: 978-989-758-351-3

CAN-bus, an inertial measurement unit and GPS. One upside to the method used was that it did not require expensive equipment. A downside, on the other hand, was an adaption phase at the start of each processed sequence, where the performance was lowered due to a lack of information. The case of long-time predictions of driver behavior was considered in (Wijnands et al., 2018). Among the behaviors studied was acceleration and whether the speed of the car was within the legal limits. The study used GPS coordinates of a test car, sampled at an interval of 30 seconds over a large period of time, in combination with other information.

The use of CNNs to classify the posture of a driver has been tested in (Yan et al., 2015). In the study images of the full body posture of the driver was used to classify four different classes: drive normal, responding to a cell phone, eating & smoking and operating the shift gear. With this method, the authors got an accuracy of 99.78% correct classifications on their dataset. The same dataset was used in (Yan et al., 2016) however instead divided into six classes: Call, Eat, Break, Wheel, Phone play, and Smoke. The classifications were made using a RCNN and achieved a mean average precision rate of 97.76%.

Optical flow fields were used in our study as in order to improve the accuracy of the CNN. This type of method was also used in (Simonyan and Zisserman, 2014). In that study, both the original images and the optical flow fields were processed in parallel and the intermediate results were concatenated at the end of the network. The study's results were better or close to the compared methods containing among others improved dense trajectories and Spatio-temporal HMAX. The results suggested that the use of optical flow gave a better performance compared to raw frames.

To the best of our knowledge there are no previous studies investigating predictions of a driver's actions inside a car with the use of neural networks. Understanding the driver's intent and thus the future state of the driver provides a possibility to further anticipate the driver's readiness to react to new conditions in a traffic situation. Hence, valuable information about how well a driver could handle a driving task or react to changes in the environment can be gained. Therefore warning systems could potentially make use of such information to provide earlier and more accurate warnings.

In (Jain et al., 2016) the method used to process facial images was based on a landmark representation of the face. It might be possible to generalize this method, from landmarks of facial images to images of the whole-body posture. Hence, more information about the driver could potentially be used as the full body posture can provide more information.

The architecture proposed in the presented study revolved around two types of neural networks, one used for classification of images and the other to predict future events based on sequences of outputs from the first network. A CNN was used for image classifications, while an RNN with LSTM generated the action intention recognition. The training and testing images for the networks depicted the whole-body posture of the driver.

2 DATA

There are currently few public datasets containing image sequences of drivers. One of the datasets was created in (Abouelnaga et al., 2017). The focus of this dataset was distracted drivers and contains the classes: Drive Safe, Talk Passenger, Text Right, Drink, Talk left, Text Left, talk Right, Adjust Radio, Hair & Makeup and Reach Behind. Unfortunately, this dataset is not on a sequential form where one class leads to another creating a chain of actions and could therefore not be used in this study. Another dataset created by (Jain et al., 2015) consists of 1180 miles of driver data which is annotated with turns, lane changes and drive straight. This dataset, even though it contains images of a driver, does not focus on driver activity. Therefore the dataset was not suitable for this study.

The dataset used in this study consists of sequences of images, collected using two cameras mounted inside of a car. During data collection, one camera was placed in the left A-pillar facing the driver and the other camera on the side window of the front passenger seat. Due to safety reasons, the car was parked during the collection of the data. There were eight participants performing 13 tasks in a Volvo V40 Cross Country. In total eight classes of driver behavior were used: drive safe, glance, lean, remove a hand from the steering wheel, reach, grab, retract and hold object. Examples of these classes can be found in Figure 1. The drivers were instructed to perform tasks in a specific order, for example, start by driving safely then glance, remove a hand from the steering wheel, lean towards the center then proceed by picking up an object. Each task was performed five times by each participant and lasted between 1.5 and 13 seconds. Around 90,000 images were collected. Due to technical difficulties, the images were captured at approximately 30 frames per second. The dataset was then divided into two separate sets where the training and testing set consisted of 55 and 10 sequences respectively for each participant. The sequences used as the test set corresponded to tasks where the driver was given very few instructions of how the task was to be performed more than that they would start by drive safe and then pick up and bring an object to them. This was to encourage a more natural approach to how the task was performed. The distributions of the classes in the training and testing set can be found in Table 1 for the CNN and in Table 2 for the RNN with LSTM. A restriction of working with sequences of actions is that different actions vary in the time it takes to perform them as well as how often they occur. It is therefore problematic to create an evenly distributed dataset. The difficulty in using normalization in order to even out the dataset is that all different types of actions can not be expected to be performed over an equal amount of time or frequency. An example is the classes glance and reach, a person might not need to glance before picking up an object but do need to reach for it. The test data was also meant to be as representative as possible of how the actions would be performed in reality, which also affects the distribution in the test set. One example is the very low number of occurrences of lean in the test dataset, which suggests that the assumption that lean would be an important class when a driver is picking up an object might not be true. It could also mean that the objects were to close to the driver for lean as a class to play a significant role. The RNN with LSTM was tested for each set of 20 frames in each sequence and evaluated on the corresponding next 20 frames unless the next 20 frames would reach outside of the sequence.

The CNN used in this study requires one ground truth class label for each image. In practice, it would be possible for more than one of the classes to be a valid label for some of the images. In order to use only one class as a label for the images, a priority system was created. The system used the last action taken by the driver that corresponded to a class as the label. The exceptions to this rule were the classes grab, retract and holding object that also had a priority over the other classes.

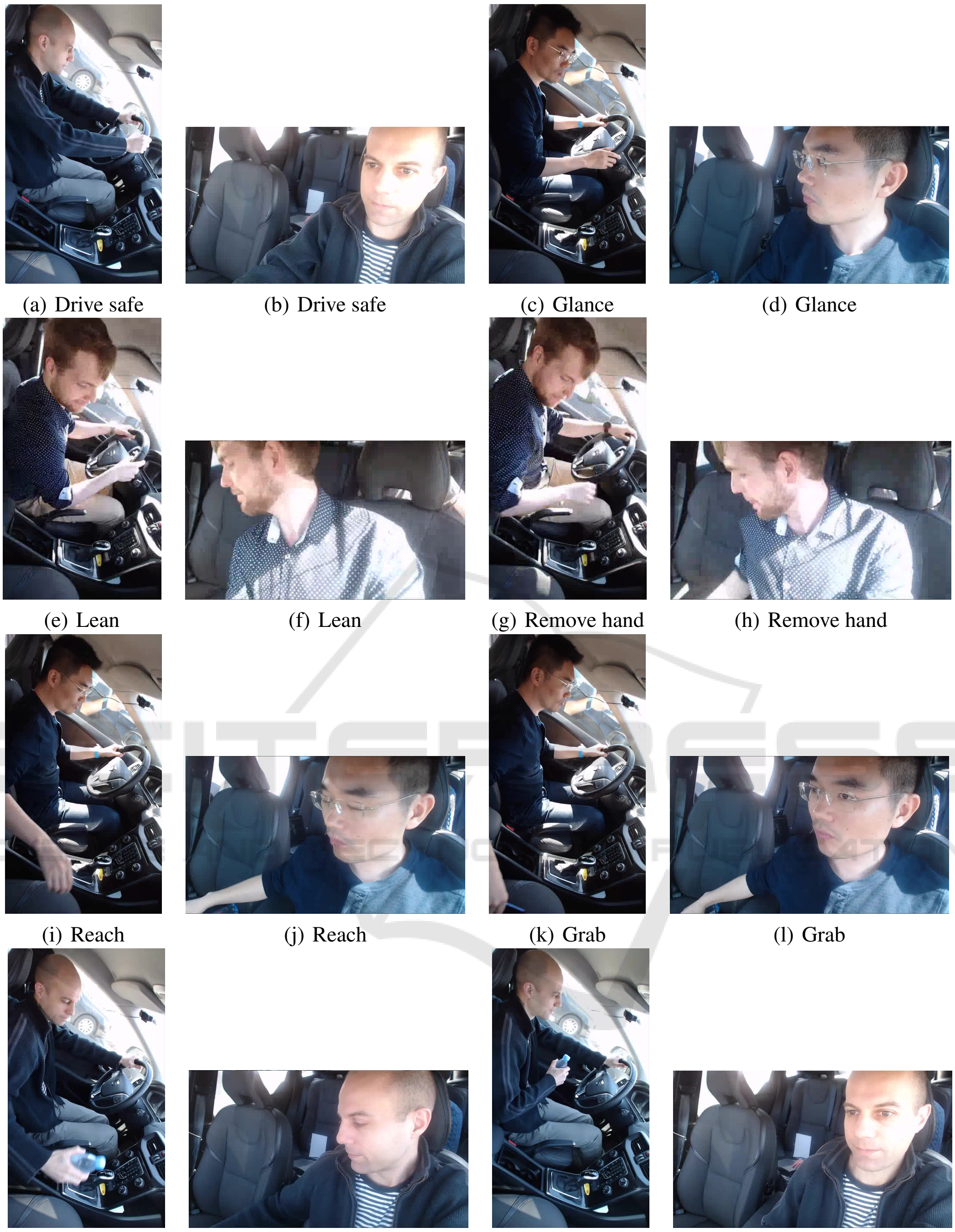
3 METHOD

3.1 Model Overview

The system proposed in this study is based on two major parts, a CNN for image classification and an RNN with LSTM for action prediction. The CNN and RNN with LSTM models used in this study were based on the work by (Torstensson, 2018). A schematic illustration of the system is shown in Figure 2, where the input to the system is a sequence of images. The first step in the processing chain is to decrease the size of the images to 256 by 128 pixels and transform them to grayscale in order to reduce the computational cost. After preprocessing, the images are classified by the CNN. The classification transformed the two-dimensional image into a scalar representing the resulting class for the given frame. The preprocessed images are also converted into optical flow fields, which are sent into the CNN in parallel to the original images. The RNN with LSTM then uses the generated sequence of classes as input to generate a sequence of scalars representing a prediction of future classes. The training process of the CNN and RNN with LSTM requires a ground truth that can map each of the images to a specific class. For benchmarking, two different inputs were given to the RNN, the ground truth of the actual behavior, and the predicted output from the CNN. Both the CNN and the RNN with LSTM used cross-entropy as the objective function.

3.2 Classification

Among the models presented in this study is a plain CNN model used for classification, which consists of two CNN blocks shown in Figure 3. The dataset was captured with two cameras, therefore each time frame provides two images. These images can be separately classified, but that can potentially lead to an increase in errors as well as not making use of the added information provided by using cameras with different angles. To make use of this data, two of the CNN blocks are used in parallel. The outputs are concatenated and followed by dense layers. As mentioned in (Yamashita et al., 2018) dropout layers can be used to reduce overfittning and have therefore been used after the dense layers. The plain CNN model does not receive any benefit from using a sequence of images compared to separate frames. Another model using dense optical flow as an additional input was created to make use of the information in the changes between the images in the sequence. Optical flow can be used as a method of representing the changes between images as a vector field. This can be done by imposing two assumptions. These assumptions are that the pixel values are preserved from one image to the other, though they might be moved around and the second one that nearby pixel values are moved at a similar rate (Fleet and Weiss, 2006). The arrays used as input to the convolution block was on the form [Batch size, Image height, Image width, Channels] where Channels was one when the grayscale images were sent in and two for the dense optical flow fields.



(a) Drive safe (b) Drive safe (c) Glance (d) Glance
(e) Lean (f) Lean (g) Remove hand (h) Remove hand
(i) Reach (j) Reach (k) Grab (l) Grab
(m) Retract (n) Retract (o) Hold (p) Hold

Figure 1: Sample images from the dataset.

The same CNN block can therefore, be used to process the optical flow fields as the CNN block processing the images and does so in parallel but with separate weights. The optical flow model can be found in Figure 4.

3.3 Prediction

One of the differences between a RNN and a CNN is that the RNN keeps a memory when processing a sequence. The elements of the network, h_t, can be calculated, given the input x_t, at a time t as described

Table 1: Class distributions in testing and training set for the CNN.

Class:	Drive safe	Glance	Lean	Remove hand	Reach	Grab	Retract	Hold object
Training:	30521	10487	4106	6492	12915	935	3228	7316.
Testing:	6035	578	28	524	2915	268	768	1884

Table 2: Class distributions in testing and training set for the RNN.

Class:	Drive safe	Glance	Lean	Remove hand	Reach	Grab	Retract	Hold object
Training:	30681	10482	4191	6493	13036	940	3233	7136
Testing:	5944	577	28	522	2915	268	766	1875

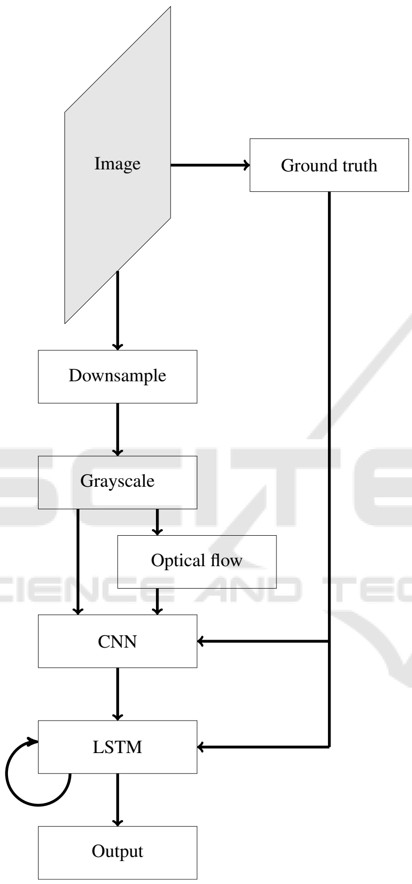


Figure 2: An illustration of the system used in this study.

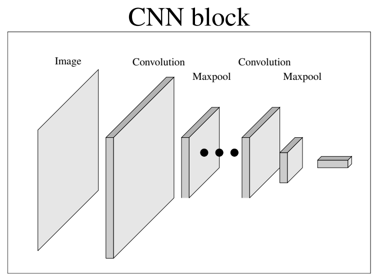


Figure 3: The structure of the CNN block.

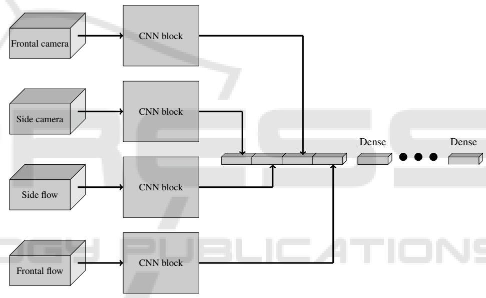


Figure 4: The structure of the CNN with optical flow.

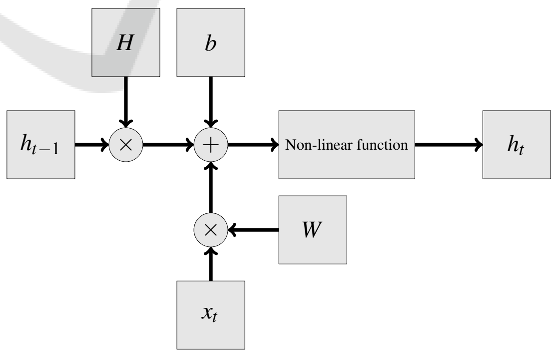


Figure 5: A chart of an RNN.

in Figure 5. Where W, H are the weights and b a bias (Jain et al., 2016).

The elements are therefore calculated by also using the previous elements, this is why it can be viewed as having a memory as previous inputs also affect the later ones. One problem with RNNs is known as the vanishing and exploding gradients problem (Pascanu et al., 2012).

One way to handle this issue is to use a version of RNN called RNN with LSTM, which is illustrated in Figure 6. The intention of adding the LSTM is, instead of storing the information in the hidden unit, to use a memory cell for information storage. The memory cell, c_t, is then passed on for each element of the sequence. The update of the hidden unit is in this case based on three gates: The input gate (i_t), the forget gate (f_t) and the output gate (o_t). These gates determine the flow of information in and out of the memory cell as well as the update of the hidden state. The σ is a logistic function, the wave is a non-linearity and x_t the input at time t. The benefit of using LSTM is that the structure uses summations, which reduces the risk of vanishing or exploding gradients (Jain et al., 2016).

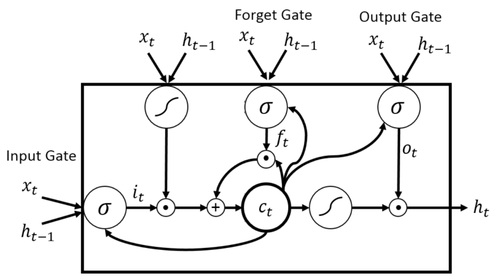


Figure 6: LSTM cell diagram from (Jain et al., 2016).

The main task of the RNN with LSTM presented in the study is to predict the next element of the input sequence. More specifically a bidirectional RNN with LSTM has been used but will be referred to as RNN with LSTM for simplicity (Schuster and Paliwal, 1997). The sequences used as input is either from the CNN or the ground truth depending on which model is being used. With a sequence of size t the predicted element can be added at $t = 0$ and the last element is removed. The newly created sequence can then be used as an input to the RNN with LSTM to generate yet another element. By repeating this process, predictions of an infinite number of frames ahead can be made. The process is illustrated in Figure 7. When one of the predictions is used as an input in the following iterations there is a risk that an incorrect prediction will lower the accuracy of coming predictions as the input is incorrect. Using this approach introduces a risk of propagating errors leading to a lowered average accuracy for each frame predicted.

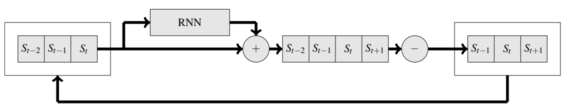


Figure 7: An overview of the prediction process of the RNN with LSTM proposed in this study.

4 RESULTS

The CNN and RNN with LSTM were trained and tested separately. The CNN was tested for different number of epochs, while the RNN was tested for different number of epochs and values of the hyperparameters. One epoch of training with the CNN using opticalflow took approximately 10 minutes on a GeForce GTX 1080 Ti. The RNN wih LSTM took slightly above one minute on the same device.

4.1 Classification

Two tests were done on the CNN, one on the plain CNN, see Section 3.2, and one on the optical flow CNN. In Figure 8 the results of the plain CNN can be seen. In this figure, the x-axis is the number of epochs trained and the y-axis the accuracy, where the accuracy is defined as the number of correctly classified images divided by the total number of images. Figure 9 shows the results of the optical flow CNN. With the first model, the best accuracy achieved was 73% while the best with the second model was 82%. The later model performed better and was the one used in later tests with the RNN.

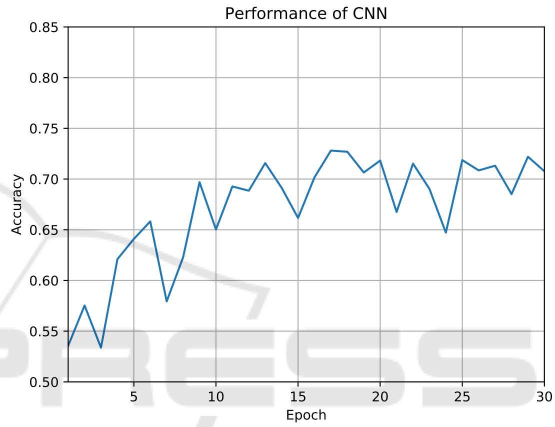


Figure 8: The accuracy of the CNN for different number of epochs trained.

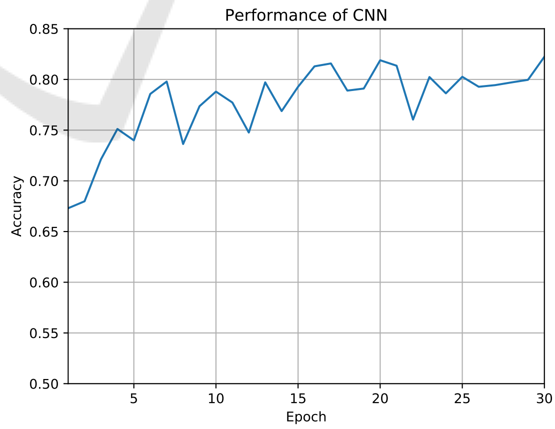


Figure 9: The accuracy of the CNN, with the addition of parallel optical flow input, for different number of epochs trained.

4.2 Prediction

In order to set the model hyperparameters of the RNN with LSTM, Bayesian optimization was used. The network was first tested with random values within set bounds of the hyperparameters. The hyperparameters, to be tuned this way, were the number of hidden units and layers, and the learning rate. In Figure 10 the results of different sets of hyperparameters can be found, the input data was from the CNN. The y-axis is the accuracy and the x-axis the number of frames predicted into the future. Similarly, the results when using the ground truth as input the performance for different sets of hyperparameters can be found in Figure 11. Testing the RNN with LSTM on the ground truth data represents a test where the CNN would perform perfectly. The best performing hyperparameters, as well as the search space set for the case of CNN input data and ground truth input data, can be found in Table 3 and 4 respectively.

Table 3: Hyperparameters used in the test of the RNN with LSTM on the CNN input data. From left to right are the hyperparameter sets shown in Figure 10, best received value, minimum bound and maximum bound.

Parameter	Set:1	Set:2	Set:3	Min	Max
Hidden units	50	148	120	50	150
Layers	2	1	2	1	2
Learningrate	0.02	0.00865	0.01610	0.00001	0.02

Table 4: Hyperparameters used in the test of the RNN with LSTM on the ground truth input data. From left to right are the hyperparameter sets shown in Figure 11, best received value, minimum bound and maximum bound.

Parameter	Set:1	Set:2	Set:3	Min	Max
Hidden units	149	50	50	50	150
Layers	1	2	2	1	2
Learningrate	0.01987	0.01423	0.01555	0.00001	0.02

The RNN with LSTM was tested both on the output from the CNN and the ground truth. Results from the tests can be found in Figures 12 and 13 respectively. The x-axis represents the number of time frames predicted into the future and the y-axis the accuracy. The accuracy is defined as the number of correct predictions divided by the total number of predictions separately for each of the time steps. The different lines are varying amounts of epochs trained and the dotted line a comparison. The intention of the comparison is to show the case if the predictions would be a repetition of the last element in the input as if there would be no transition between classes.

The best results with the CNN input data was 84% accuracy when predicting one frame ahead and 58% accuracy when predicting 20 frames ahead. The network outperformed the comparison for all future time steps with a growing margin for longer time predictions. In the case when using the ground truth data as input, the best performance gave an accuracy of 96% after one frame and 61% after 20 frames. The

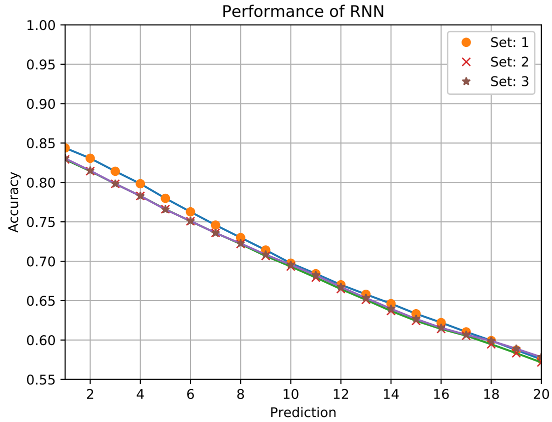


Figure 10: Accuracy of the RNN with LSTM, tested on the CNN input data, at different numbers of frames predicted. Each line represents one set of hyperparameters, which can be founds in Table 3.

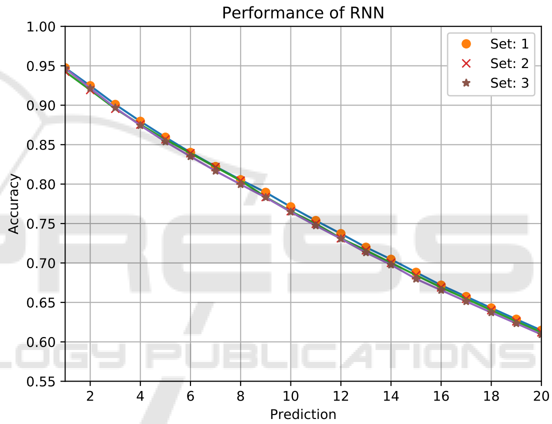


Figure 11: Accuracy of the RNN with LSTM, tested on the ground truth input data, at different numbers of frames predicted. Each line represents one set of hyperparameters, which can be found in Table 4.

comparison performed better for very short time predictions but fell of quicker resulting in a better performance of the network with longer time predictions. There is a clear trend in both cases of dropping accuracy with longer time predictions. This could probably, at least in part, be related to the accumulating error in the model used.

To further evaluate the performance of the RNN with LSTM, trained and tested with ground truth data as input, confusion matrices were created containing the recall and precision as defined in (Powers, 2008). The results for one frame predictions can be seen in Tables 5 and 6, while the 20 frames predictions can be found in Tables 7 and 8.

Out of all classes, the ones that the model performed best for were drive safe, reach and hold. These classes were also the ones appearing the most in the

Table 5: Recall of one frames prediction.

	Ground truth							
Predicted	Drive safe	Glance	Lean	Remove hand	Reach	Grab	Retract	Hold
Drive safe	1.0	0.07	0.04	0.09	0.0	0.0	0.0	0.0
Glance	0.0	0.92	0.04	0.06	0.0	0.0	0.0	0.0
Lean	0.0	0.0	0.79	0.0	0.0	0.0	0.0	0.0
Remove hand	0.0	0.01	0.07	0.84	0.02	0.0	0.0	0.0
Reach	0.0	0.0	0.07	0.0	0.97	0.27	0.0	0.0
Grab	0.0	0.0	0.0	0.0	0.0	0.27	0.0	0.0
Retract	0.0	0.0	0.0	0.0	0.0	0.42	0.81	0.03
Hold	0.0	0.0	0.0	0.0	0.0	0.03	0.19	0.97

Table 6: Precision of one frames prediction.

	Ground truth							
Predicted	Drive safe	Glance	Lean	Remove hand	Reach	Grab	Retract	Hold
Drive safe	0.98	0.01	0.0	0.01	0.0	0.0	0.0	0.0
Glance	0.02	0.92	0.0	0.05	0.01	0.0	0.0	0.0
Lean	0.0	0.0	0.88	0.08	0.0	0.04	0.0	0.0
Remove hand	0.0	0.01	0.0	0.83	0.14	0.0	0.0	0.0
Reach	0.0	0.0	0.0	0.0	0.98	0.02	0.0	0.0
Grab	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
Retract	0.0	0.0	0.0	0.0	0.0	0.16	0.8	0.04
Hold	0.0	0.0	0.0	0.0	0.0	0.01	0.14	0.85

dataset, while the ones appearing significantly less could obtain a value as low as 0%.

5 DISCUSSION

One possible reason for why the CNN could be improved with optical flow is that it works in the time dimension as well. Hence, information in the sequence which is hard to extract from single frames, such as the direction of movement, can be used. This effect could also potentially be useful with the priority system used for annotating the data. One still frame might not be sufficient to extract the information of which action was performed last.

The results of the best performing hyperparameters for the RNNs with LSTM showed that different sets of hyperparameters performed similarly. It appeared that the best choice of the number of layers was one for the ground truth input and two for the CNN input. The best value of the learning rate was also high, which might have been due to the low amount of epochs used in each training phase.

The results in (Jain et al., 2016) and (Pech et al., 2014) are difficult to compare with the results in this study due to the differences in timescale and actual task. The predictions focused on movements of the car rather than the actions of the driver. Consequently the timescale differs from the one in this study where the classes changes constantly during a sequence and a single class can last a few frames. The sensor fusion approach used in (Jain et al., 2016) could provide an interesting prospect of further work if combined with the method in the study presented here.

The studies (Yan et al., 2015) and (Yan et al., 2016) achieved higher classification accuracy on their dataset than the one presented here. A major difference between these studies and the one presented here is that our study classifies sequences of images with shifting classes, which the other studies, to our knowledge, does not. Therefore there are transitions between the classes, which can be challenging for a machine or human to classify and therefore contributing to the difference in classification accuracy.

The variation in performance, on different classes, shown by the confusion matrices could be due to how the more well-represented classes become more likely to be predicted as they have a larger probability of appearing. With an increasing amount of the smaller classes becoming part of the larger classes in each iteration the probability of prediction might increase for the larger classes as they appear more often. This type of effect could potentially be mitigated with the use of a bias correction that evens out the probability of each class being chosen. This type of correction, on the other hand, run the risk of creating a network

Table 7: Recall of 20 frames prediction.

		Ground truth							
		Drive safe	Glance	Lean	Remove hand	Reach	Grab	Retract	Hold
Predicted	Drive safe	0.94	0.77	0.54	0.79	0.28	0.0	0.0	0.0
	Glance	0.05	0.19	0.46	0.08	0.09	0.0	0.0	0.0
	Lean	0.0	0.0	0.0	0.0	0.01	0.0	0.0	0.0
	Remove hand	0.01	0.03	0.0	0.13	0.12	0.04	0.02	0.0
	Reach	0.01	0.0	0.0	0.0	0.51	0.96	0.9	0.23
	Grab	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Retract	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.01
	Hold	0.0	0.0	0.0	0.0	0.01	0.0	0.08	0.76

Table 8: Precision of 20 frames prediction.

		Ground truth							
		Drive safe	Glance	Lean	Remove hand	Reach	Grab	Retract	Hold
Predicted	Drive safe	0.65	0.09	0.0	0.07	0.18	0.0	0.0	0.0
	Glance	0.29	0.18	0.02	0.06	0.45	0.0	0.0	0.0
	Lean	0.0	0.0	0.0	0.0	0.95	0.0	0.0	0.05
	Remove hand	0.04	0.04	0.0	0.11	0.74	0.02	0.04	0.01
	Reach	0.01	0.0	0.0	0.0	0.51	0.09	0.24	0.15
	Grab	nan	nan	nan	nan	nan	nan	nan	nan
	Retract	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
	Hold	0.0	0.0	0.0	0.0	0.01	0.0	0.04	0.95

that repeatedly predicts small classes that barely ever appear and therefore lowers the accuracy. The precision and recall could potentially also be improved by using a cost function which uses those measurements. By changing the cost function there is potential to reward the correct classification of the smaller classes and better adjust to the distribution of the dataset.

It would also be possible to change the dataset from the system of priority into a system where several classes can be used as a label for an image. One potential benefit of this could be an increase in the accuracy of the CNN with less reliance on the optical flow. Hierarchical clustering is a method that potentially could be used. If done in a divisive manner one general class could be split up into subclasses, which in turn also can be split up into further subclasses. In such a manner a tree structure of classes could be created (Zhang et al., 2017).

The CNN used for classification could potentially be improved by implementing features from some of the networks performing well in benchmarks such as ImageNet, one example is GoogleNet (Russakovsky et al., 2015). The network used could also easily be replaced with another network trained for the same type of data as the CNN was used as a separate module.

6 CONCLUSIONS

This work presents an RNN with LSTM to predict action and intention for car drivers. The proposed system consists of two modules: one CNN and one RNN with LSTM. The CNN classifies images into one of the eight known action classes. The CNN use four parallel images as input, two with images from two different cameras mounted inside the cabin of the test car and two images that correspond to the optical flow fields of the two camera images.

The classification accuracy of the CNN was 73% while using only the two camera images and when adding the two optical flow images 82% accuracy was achieved. The RNN with LSTM was used to predict future intentions of the car driver. The RNN with LSTM takes as input an array of classifications made by the CNN to predict the action in the next frame. Two approaches were compared. One where the RNN with LSTM classifications from the CNN were used as input and one where the ground truth was used as input.

The best prediction accuracies achieved was 84% for one frame ahead predictions and 58% after 20 frames ahead in the case of using the output of the CNN as input data. When using the ground truth as input the accuracies were increased to 96% after one

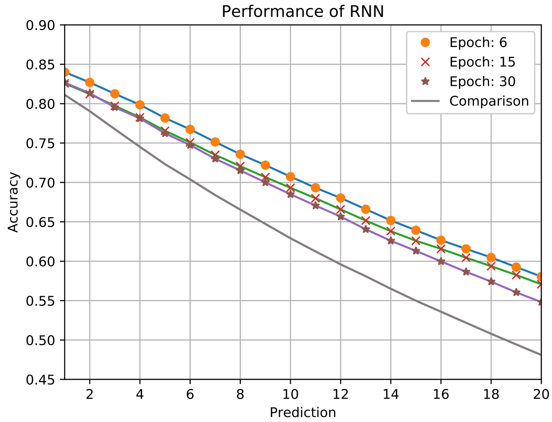


Figure 12: Accuracy of the RNN with LSTM with the best performing hyperparameters, tested on the CNN input data, at different numbers of frames predicted. Each line represents a different amount of epochs trained and the dotted line is a comparison.

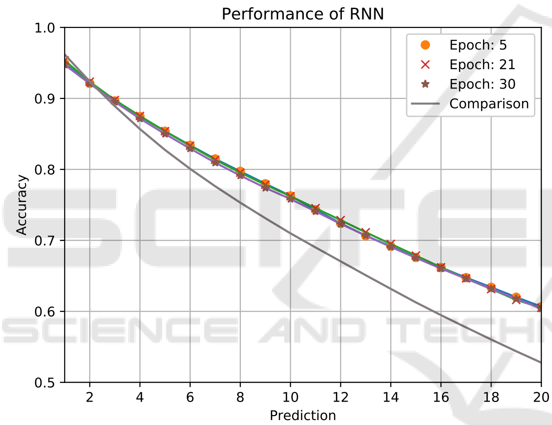


Figure 13: Accuracy of the RNN with LSTM with the best performing hyperparameters, tested on the ground truth input data, at different numbers of frames predicted. Each line represents a different amount of epochs trained and the dotted line is a comparison.

frame ahead and 61% after 20 frames.

ACKNOWLEDGMENT

This work is part of the AIR project (action and intention recognition in human interaction with autonomous systems), financed by the KK foundation under the grant agreement number 20140220.

REFERENCES

Abouelnaga, Y., Eraqi, H. M., and Moustafa, M. N. (2017). Real-time distracted driver posture classification. *CoRR*, abs/1706.09498.

Carmona, J., Garca, F., Martn, D., Escalera, A. d. l., and Armingol, J. M. (2015). Data fusion for driver behaviour analysis. *Sensors*, 15(10):25968–25991.

Fleet, D. and Weiss, Y. (2006). *Optical Flow Estimation*, pages 237–257. Springer US, Boston, MA.

Jain, A., Koppula, H. S., Raghavan, B., Soh, S., and Saxena, A. (2015). Car that knows before you do: Anticipating maneuvers via learning temporal driving models. In *The IEEE International Conference on Computer Vision (ICCV)*.

Jain, A., Singh, A., Koppula, H. S., Soh, S., and Saxena, A. (2016). Recurrent neural networks for driver activity anticipation via sensory-fusion architecture. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3118–3125.

Olabiyi, O., Martinson, E., Chintalapudi, V., and Guo, R. (2017). Driver Action Prediction Using Deep (Bidirectional) Recurrent Neural Network. *ArXiv e-prints*.

Pascanu, R., Mikolov, T., and Bengio, Y. (2012). Understanding the exploding gradient problem. *CoRR*, abs/1211.5063.

Pech, T., Lindner, P., and Wanielik, G. (2014). Head tracking based glance area estimation for driver behaviour modelling during lane change execution. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 655–660.

Powers, D. (2008). Evaluation: From precision, recall and f-factor to roc, informedness, markedness & correlation. *Mach. Learn. Technol.*, 2.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Simonyan, K. and Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 568–576. Curran Associates, Inc.

Singh, S. (2015). Critical reasons for crashes investigated in the national motor vehicle crash causation survey. Washington, DC: National Highway Traffic Safety Administration. (Traffic Safety Facts CrashStats. Report No. DOT HS 812 115).

Torstensson, M. (2018). Prediction of driver actions with long short-term memory recurrent neural networks. Master's thesis, Chalmers University of Technology. Retrieved from http://studentarbeten.chalmers.se/.

Wijnands, J. S., Thompson, J., Aschwanden, G. D., and Stevenson, M. (2018). Identifying behavioural change among drivers using long short-term memory recurrent neural networks. *Transportation Research Part F: Traffic Psychology and Behaviour*, 53:34 – 49.

World Health Organization (WHO) (2015). *Global Status Report on Road Safety 2015*. WHO Press, Geneva.

Yamashita, R., Nishio, M., Do, R. K. G., and Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9(4):611–629.

Yan, C., Zhang, B., and Coenen, F. (2015). Driving posture recognition by convolutional neural networks. In *2015 11th International Conference on Natural Computation (ICNC)*, pages 680–685.

Yan, S., Teng, Y., Smith, J. S., and Zhang, B. (2016). Driver behavior recognition based on deep convolutional neural networks. In *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pages 636–641.

Zhang, Z., Murtagh, F., Van Poucke, S., Lin, S., and Lan, P. (2017). Hierarchical cluster analysis in clinical research with heterogeneous study population: highlighting its visualization with r. *Annals of translational medicine*, 5(4):75.