2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

Multimodal Spatio-Temporal Information in End-to-End Networks for Automotive Steering Prediction

Mohamed Abou-Hussein[1], Stefan H. Müller[2] and Joschka Boedecker[3]

Abstract— **We study the end-to-end steering problem using visual input data from an onboard vehicle camera. An empirical comparison between spatial, spatio-temporal and multimodal models is performed assessing each concept's performance from two points of evaluation. First, how close the model is in predicting and imitating a real-life driver's behavior, second, the smoothness of the predicted steering command. The latter is a newly proposed metric. Building on our results, we propose a new recurrent multimodal model. The suggested model has been tested on a custom dataset recorded by BMW, as well as the public dataset provided by Udacity. Results show that it outperforms previously released scores. Further, a steering correction concept from off-lane driving through the inclusion of correction frames is presented. We show that our suggestion leads to promising results empirically.**

Index Terms— **Autonomous steering, deep learning, spatio-temporal model, multimodal input, optical flow, RNN-LSTM**

I. Introduction

Autonomously driven vehicles are seen as one major potential solution for multiple issues in transportation. Having safer roads, reducing road congestion and increasing one's mobility are all examples of these potential advantages. The classical autonomous driving solutions have an encode-decode pipeline process of two modules, the perception module and the control module. The former encodes the visual data based on human-selected features and feeds it to the latter which decodes the representation to estimate the steering control based on defined rules [1]–[4]. However, end-to-end steering couples the whole pipeline process in a single module that performs the encoding-decoding process through deep learning. Further, advances in deep learning for computer vision [5, 6] motivates the use of visual data in an end-to-end approach within the scope of autonomous driving.

This research proposes a neural network based vehicle steering controller trained in an end-to-end fashion from the camera images to regress the steering command. This concept will arguably lead to better steering performance in comparison to classical steering techniques, which are dependent on hand-crafted features from the camera input. The trained deep learning model performs the automatic feature extraction of the important criteria more efficiently and generically [7]. Additionally, this will lead to an overall smaller system model [8]. The scope of this paper is limited to lane keeping. while previous research presented notable advances [8]–[11], what is not clear from the studies is the importance of different modalities, the significance of time context and the traversability of the prediction. Henceforth, We carry out an empirical analysis between spatial, spatio-temporal and multimodal concepts to assess the performance of each. Precedent researches used driving error deviation as the main metric of evaluation. We extend the formerly utilized evaluation metric and include an assessment for the traversability of the steering using a proposed metric score for smoothness. The study of the results prompted a proposal of a multimodal recurrent model that uses spatial information and optical flow as input. The suggested model outperforms previous concepts. We validate our proposal through the use of the open-source dataset provided by Udacity[1] and show that it outperforms state-of-the-art results. Furthermore, we extend the training data with driving recovery images and show the validity of the concept in handling off-course driving situations.

II. Related Work

The end-to-end approach for autonomous steering was previously discussed in literature. The work by Nvidia [8] introduces a fairly simple deep convolutional neural network (CNN) that extracts features automatically by convolutional layers followed by a sequence of fully connected layers to regress the steering angle using a large dataset composed of almost 100 hours of driving, also the study was limited to lane keeping only. Simulated and on-road tests were performed and the model was found to rarely fail according to their evaluation metric. The work in [12] alters the problem from a regression to a classification one. This is done by discretizing the steering domain into multiple logarithmic or linear bins. Thus, the supervised learning fully convolutional network-long short term memory (FCN-LSTM) model is used to predict which range of steering is correct. The results exhibit the capabilities of deep learning in scene interpretation and understanding. For example, in cases where the vehicle road has some sort of an obstacle straight in front of the vehicle, the lowest given steering probabilities were around the angle $0°$.

Udacity organized a challenge where several teams proposed their solution for an open source dataset[2] and a

[1]Mohamed Abou-Hussein, Graduate of Computer Science, Freiburg University, Georges-Köhler-Allee 080, 79110 Freiburg im Breisgau – mohamed.abou-hussein@informatik.uni-freiburg.de
[2]Stefan H. Müller, BMW Group, Landshuter Straße 26, 85716 Unterschleiheim – stefan.sh.mueller@bmw.de
[3]Joschka Boedecker, Neurorobotics Lab, University of Freiburg, and BrainLinks-BrainTools, Georges-Köhler-Allee 080, 79110 Freiburg im Breisgau – jboedeck@informatik.uni-freiburg.de

[1]https://www.udacity.com/self-driving-car
[2]https://www.github.com/udacity/self-driving-car

leaderboard of the best results was posted. Other work on spatial neural networks was done by [10] through transfer learning using a pre-trained ResNet-50 [6] and by [13] through proposing a simple deep CNN. Both architectures yielded promising results in imitating the driver's behavior. Moreover, the work by [10, 11, 13] examined the problem with spatio-temporal architectures, which are models that would not only encode the visual data from a single frame but also encode the relation between the different frames. For example, models such as a recurrent neural network (RNN) with an LSTM cell were found to boost the performance of the network in most cases. Moreover, adopting the concept of multi-modality was also reported to enhance the performance. For instance, the work in [11] coupled the loss with the torque and the speed while the work in [13] incorporated the speed in its model input. Previous research focuses on generating a model that imitates the learned behavior from the supervised data. However, it mostly lacks assessment of the smoothness of the steering command regressed. A continuous-smooth command is critical so as not to have a zigzag behavior and to provide a safe and convenient driving for the passengers. Thus, further analysis will be done in regard to this matter in this research.

The VisualBackProp algorithm developed in [14] was used to validate which parts of the input image activate the network, enabling an understanding of which features are selected by the network. The work in [7] shows that a simple CNN is able to encode the important elements of the frame and uses it to compute the command of the steering angle. This study shows that end-to-end learning with the right dataset is able to automatically extract the important features from the visual data directly.

The concept of having multiple input modalities through the inclusion of explicit motion information with the spatial information was used in [15]–[17]. The context of the aforementioned research was image sequence to text generation. The explicit motion information is represented by the optical flow between consecutive frames [18, 19]. Results show that this concept has outperformed strictly spatial CNN models as well as single modal RNN. Thereby, we study this concept within the scope of autonomous steering.

III. Datasets

The work in this thesis utilizes two sets, the BMW dataset and the open-source Udacity dataset. Figure 1 shows scattered samples of both sets. We use the former for the empirical analysis of our approach and the latter to validate our findings against previous research.

A. The BMW dataset

This dataset is provided internally by BMW and is not available for public use. The data is collected from an onboard vehicle camera on one of the BMW models while driving mainly on highway roads. The decoded RGB front camera frames are of the size $960 \times 540 \times 3$. No extreme weather conditions were available in the dataset. Only lane keeping frames were used. The data was split into training

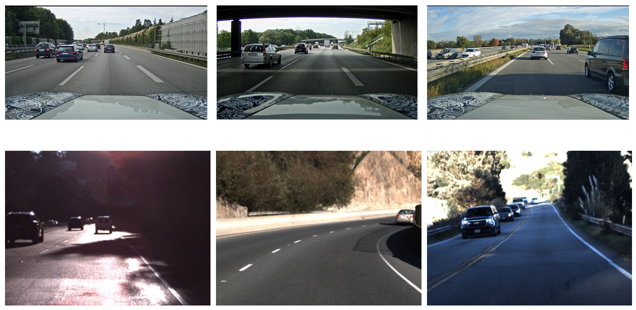


Fig. 1: Samples from the BMW dataset (top row) and the Udacity dataset (bottom row).

and validation subsets of $140,201$ and $11,425$ frames, respectively. The steering data takes values between $-10°$ and $10°$, with a mean of $0.11°$ and a standard deviation of $1.90°$ and $2.15°$ for the training and validation sets, respectively. Vehicle speed data yielded relatively high means of 106.76 km/h and 132.64 km/h for the training and the validation datasets, respectively.

B. The Udacity dataset

The dataset consists of $33,808$ frames for training and $5,614$ frames for testing of the size $640 \times 480 \times 3$ (RGB). The Udacity dataset has a higher variance in the lighting conditions than the BMW dataset. Specific details about the road conditions can be found by the dataset publishers[3]. In contrast to the BMW set, the speed of the vehicle has a much lower mean value of 15.63 km/h for the training set. Further, the steering angles takes higher values between $-50°$ and $50°$ with means of $-0.49°$ and $0.15°$ and with standard deviations of $15.6°$ and $11.90°$ for the training and the validation sets, respectively.

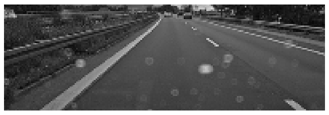
IV. Definition and Approach

Given a series of frames from the vehicle on-board camera $X_t = \{x_1, x_2, .., x_{N-1}, x_N\}$, we compute $\hat{X}_t = \{\hat{x}_1, \hat{x}_2, .., \hat{x}_{N-1}, \hat{x}_N\}$ such that $\hat{x}_t$ is the processed data format of x_t. Similarly, given the recorded control steering angles of the vehicle $Y_t = \{y_1, y_2, .., y_{N-1}, y_N\}$ at the same time instances t of X_t, the problem can be defined as estimating the function F mapping $\hat{x}_t \rightarrow y_t$.

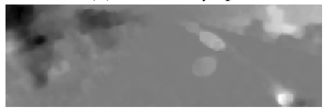
A. Data preprocessing

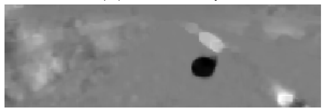
Initially, we decode the BMW data frames into the RGB format with the size of $960 \times 540 \times 3$. We crop the hood and the sky as they contain no information and resize the frames to $200 \times 66 \times 3$. Tests on the BMW dataset showed no significant differences between results for the YUV and the gray-scale color-spaces. Therefore, we use the gray-scale format. The work by Nvidia mentions that each frame is normalized. However, the normalization details were not stated. We perform image normalization using subtraction of the mean and dividing by the standard deviation. In the cases

[3]https://github.com/udacity/self-driving-car/tree/master/datasets/CH2


(a) Frame x_{t-1}.

(b) Frame x_t.


(c) Optical flow horizontal component.


(d) Optical flow vertical component.

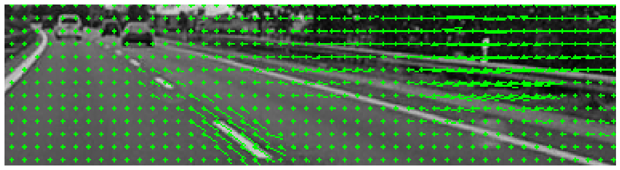

(e) Optical flow visualization.

Fig. 2: (a) and (b) two consecutive frames. (c) and (d) horizontal and vertical components of the optical flow. The input to the model is the stacked tensor of (b), (c) and (d). Brighter/darker spots relate to segments where more motion is captured by the algorithm. (e) magnified area of (b) showing the optical flow. The motion of lane road markers and the side road borders is captured.

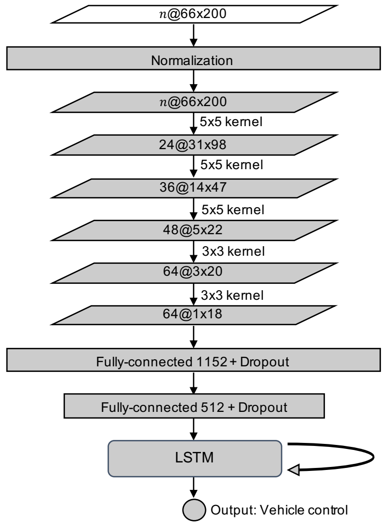


Fig. 3: Our proposed model for an RNN-LSTM. n represents the input channels and is dependant on the variation of the input type (spatial or multimodal).

where the input consists of multiple modalities, such as that described in Subsection IV-B.4, we apply the normalization on each modal input separately and stack the normalized inputs to form a single 3D tensor. No data augmentation was implemented.

B. Methods

1) Convolutional Neural Network (CNN): As a starting point, we replicate the state-of-the-art model proposed by Nvidia [8] and perform minor optimizations for its hyper-parameters. The model's top 5 layers are convolutional with the first 3 layers having kernels of size 5×5 and a stride value of 2, followed by 2 layers of 3×3 kernels and a stride value of 1. The layer sizes are 24, 36, 48, 64 and 64. Four fully connected layers of sizes 1164, 100, 50 and 10 follow. The input frame shape is of the size 200×66.

2) 3D Convolutional Neural Network (3D-CNN): The concept of 3D-CNN [20] for spatio-temporal applications has shown that it leads to better features extraction [21]. It is based on stack feeding of multiple consecutive frames to the model as a single input. The concept aims at providing the network with *implicit* information about the motion and driving dynamics through which it can compute a more accurate steering command. We stack 3 sequential gray-scale frames of size 200×66 on top of each other to form a 3D tensor. We use the same model proposed by Nvidia but replace the 2D kernels by 3D kernels of the same width and height but a depth of 3.

3) CNN with Image Difference as Input (spatial + difference): This spatio-temporal concept is based on providing motion difference between the frames to the model. The concept aims at providing *explicit* information about the motion rather than just providing the data and depending on the training of the model to compute it implicitly, such as in the 3D-CNN. The difference frame is computed as $\Delta x_t = \frac{x_t - x_{t-1}}{2} + 128$. We stack the frames x_t and Δx_t to form a 3D tensor of size $200 \times 66 \times 2$. We use the Nvidia model with 2 input channels and the rest of the model specifications are not altered.

4) Multiple input modality using optical flow (spatial + optical flow): Similar to the previous concept, the target is to feed the model explicit information about the motion as input. Thereby, we compute the normalized optical flow between the current and the previous frame and stack it with the current gray-scale frame. The optical flow represents the displacement field of the pixels between 2 given frames [18]. Within this research, we compute dense optical flow using Farneback's algorithm [19]. The term *dense* denotes computing the optical flow for all the pixels. Applying the flow algorithm yields two 2D tensors of the same size of the original image, one tensor representing the vertical components and the other representing the horizontal components (Figure 2).

5) Recurrent Neural Network - Long Short-Term Memory architecture (RNN-LSTM): Fixed input size and assumption of input independence limits conventional CNNs and motivates the use of RNNs. A recurrent relation is fed from the previous hidden layer h_{t-1} to h_t. We expect that the use of a RNN would actually lead to better smoothness scores due to the fact that the error is computed across the whole given training sequence[4]. Thereby, the error in the gradient of the steering angle is propagated and the model should

[4]While smoothing with low-pass filters can be used in post-processing, this has the disadvantages of introducing delays, potential instabilities and requiring to hand-pick the filter parameters. In RNNs this filtering is automatically learned.

Model	Input variation	Input shape	L_{MSE}	Smoothness (γ)
Target (benchmark)	–	–	–	0.088
Constant 0° steering	–	–	0.001420	–
CNN (Nvidia)	spatial	$1@200\times66$	0.000541	0.281
3D-CNN	stacked sequence	$3@200\times66$	0.000471	0.317
CNN (Nvidia)	spatial + difference	$2@200\times66$	0.000460	0.318
CNN (Nvidia)	spatial + optical flow	$3@200\times66$	0.000330	0.402
RNN-LSTM	spatial	$1@200\times66$	0.000441	0.121
Multimodal RNN-LSTM	spatial + optical flow	$3@200\times66$	0.000247	0.126
Multimodal RNN-LSTM (HPO)	spatial + optical flow	$3@200\times66$	**0.000215**	**0.106**

TABLE I: Compiled error and Smoothness scores for the tested concepts on the BMW data. Scores are averages of 3 runs.

learn to smooth the output in a similar manner to the input sequence smoothness. Figure 3 shows our proposed RNN consisting of the same 5 convolutional layers with the same kernel sizes and strides as Nvidia's model. The use of the same convolutional layers design is motivated by a follow-up research work that showed that this convolutional design extracts the important features excellently [7]. The encoded data is then flattened in 2 consecutive fully connected layers of size 1152 and 512, respectively, followed by an LSTM cell with 100 hidden units whose output is fully connected to 1 single output neuron. The choice of the LSTM cell was motivated to handle the problem of vanishing gradients [22]. The fully connected layers were empirically designed.

6) Multimodal RNN-LSTM: This concept is motivated by previous research [15]–[17] and the experiments results as to be shown in Section V. We combine two variations to yield a recurrent architecture (Figure 3) with an input of multiple modalities (Figure 2).

C. Evaluation metrics

We identify two metrics for the evaluation of our approaches.

1) Steering Error Metric: This is the main metric used for evaluation of the steering models. We aim to analyze how much is the mean deviation between the regressed steering angle and the actual recorded data. For this purpose, we minimize the MSE error loss:

$$L_{\text{MSE}} = \frac{\sum_1^N (y_t - \hat{y}_t)^2}{N} \tag{1}$$

2) Steering Smoothness Metric: We propose a smoothness metric γ such that:

$$\gamma = \sigma(\hat{Y}_t - \hat{Y}_{t-1}) \tag{2}$$

where σ is the standard deviation operator applied to the time series of the difference between the regressed outputs $\hat{Y}_t$ and the previous regressed output at $\hat{Y}_{t-1}$. In concept, we are computing the standard deviation of the gradients of the change of the discrete steering values. This metric is derived from our assumption that locally, the difference between the steering angles for a smooth steering is small. A smaller value of γ indicates less variation in the gradient, hence smoother steering. Note that this metric is not incorporated in the loss function.

D. Training configurations

All the experiments were executed using the Tensorflow framework [23]. Different input variations are run on the CNN model for 32 epochs. The empirical evaluation showed signs of overfitting. Thus, we regularize the model with a dropout of 85%. This is applied over all the fully connected layers. Each variation is trained 3 times and the scores were computed through averaging all the runs. We train the model with a fixed batch size of 100. ADAM optimizer [24] is used as the optimization algorithm with its Tensorflow default parameters of $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$ and a learning rate of 10^{-4}. For the RNN-LSTM, we apply the same configuration. The size of each sequence example for training is 10, the batch size is 10 and we use an LSTM cell of 100 hidden units.

V. Results

The results in Table I confirm our hypothesis of dealing with the problem from a spatio-temporal perspective rather than just the spatial one. All concepts that incorporated information about previous states scored a lower L_{MSE} than the strictly spatial approach. Furthermore, The **spatial + optical flow** and the **spatial + difference** variations scored better results than just stacking the images in the **3D-CNN**. This shows that explicitly feeding temporal information to the model is beneficial. In regard to the smoothness scores, all non-recurrent models scored a worse γ in comparison to the target score of 0.088. However, the proposed RNN-LSTM model achieved a significantly better γ in comparison to other non-recurrent models. This supports our hypothesis of the smoothness learning. Additionally, the RNN-LSTM L_{MSE} yielded also an improvement from the strictly spatial CNN. Yet, the L_{MSE} of other multiple input modalities variations was either equivalent or better. The multimodal RNN-LSTM model which is based on the combination of the two variations which has the best steering L_{MSE} and γ scores successfully reduces the steering error score to $L_{\text{MSE}} = 0.000247$, which is better than all previous concepts. Furthermore, it achieves the second best $\gamma = 0.126$.

We apply the optical flow algorithm on the cropped version of the images. We analyze the applicability of the algorithm for online applications by executing the algorithm sequentially for 200 iterations and computing the average run-time which yielded 0.0431 seconds. This makes the algorithm feasible for online applications roughly under 23

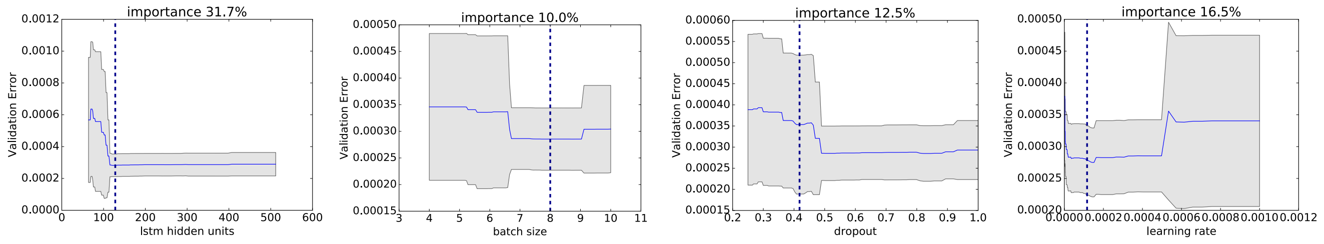


Fig. 4: Marginal perplexity recorded for changing values for each parameter across the x-axis is shown in each plot. The solid blue line and the gray fill shows the predicted marginals (mean one standard deviation). The dashed dark blue line shows the best run value for each parameter.

frames per second (FPS). The hardware used for the pre-processing is an Intel i7 sixth generation CPU. Comparable autonomous hardware toolkit is developed and used in autonomous vehicle technology research [25]. Also, the BMW and the Udacity datasets have rates of 14 and 20 FPS respectively. The aforementioned shows the online run-time feasibility of the algorithm used.

A. *Hyper-parameter optimization (HPO)*

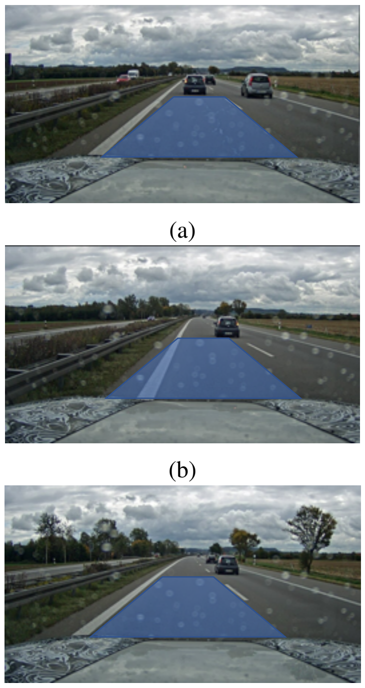
We extend the study and perform hyper-parameter optimization for some of the problem parameters to further improve the results. We define 4 hyper-parameters as our configuration space. These are the learning rate, the batch size, the dropout and the LSTM hidden units size. We use the Hyperband Bayesian Optimization (BO-HB) approach [26]. The optimizer runs 73 iterations in total, each with a different configuration. Table II shows the range of the hyper-parameters specified to the optimizer as well as the best value for each parameter. We re-run the multimodal RNN-LSTM with the recorded optimized hyper-parameters. This results in the minimum $L_{\text{MSE}} = \mathbf{0.000215}$ and $\gamma = \mathbf{0.106}$. In the same context, we apply a functional analysis of variance (FANOVA) framework [27] that assesses the significance of each hyper-parameter to the problem. Figure 4 shows the results for our parameters. As listed on the graphs, the number of hidden units in the LSTM and the learning rate represent the most important parameters with 31.7% and 16.5% respectively. It is worth mentioning that several other recurrent concepts have been tested. These concepts are the **Vanilla-RNN**, the **multi-stream RNN** (similar to the work in [15]–[17]) and the **RNN with skip connections**. No improvement in the results was recorded (results not shown).

Parameter	lower bound	upper bound	best value
learning rate	10^{-6}	10^{-3}	1.16×10^{-4}
batch size	4	10	8
dropout	0.25	1.0	0.41
lstm hidden units	64	512	128

TABLE II: Hyper-parameters.

B. *Recovering from Off-lane Driving*

Lacking a realistic simulator, it is difficult to decide whether the network's steering would be good enough for a human passenger. The work in [8, 10] suggest synthetic frames shifting in the data to simulate the error correction process in the dataset during training. In the same context, we introduce and test a new approach for error correction using imitation learning. The BMW dataset included multiple instances of undesired driving behavior where the driver would steer the vehicle out and back to the center of the lane. The sequence of frames that contain this behavior was divided into two segments, the driving out-of-lane segment and the driving back-in (Figure 5). Only the latter segment is included in the training set, but both segments were included in the validation dataset to assess the model behavior. We include 14 correction instances, 8 instances are steering out to the left and 6 instances to the right. The validation dataset includes 6 scenarios divided equally left and right.


(a)

(b)

(c)

Fig. 5: Illustration of one of the steering recovering segments. (a) and (b) enclose the frames of steering out-of-lane. (b) and (c) enclose the recovery frames.

The recorded steering data of the out-and-back in lane driving looks roughly like a sinusoidal wave when plotted

(Figure 6). In all the scenarios, the driver aggressively takes an out-of-lane steering action. Once the vehicle gets off-road, the model automatically starts to predict the opposite steering angle from that of the driver. It follows that the model output roughly mimics the driver's behavior in correcting the vehicle's direction to the center of the lane. This shows that the model is able to learn to perform off-lane driving recovery.

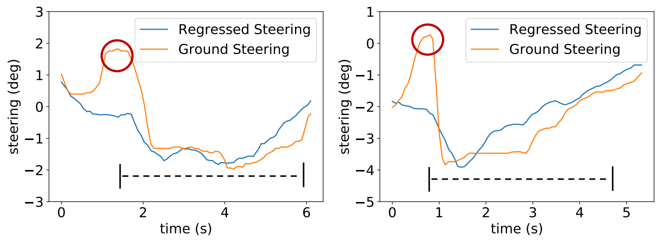


Fig. 6: Steering correction segments showing recovery behavior of the trained model. The circle represents the moment the driver starts to take the corrective action. The segment of correction by the driver is marked with the dashed line.

C. Udacity dataset results

We train the RNN-LSTM multimodal model with the optimized hyper-parameters on the Udacity dataset to relate our findings with other research. We apply the same data preprocessing procedure. The only difference is the cropped area. Following the same concept, we omit the upper part of the image as it contains no important information. We crop the top 268 pixels to have a 640×212 and resize it to 200×66. We report the $L_{\text{RMSE}} = \sqrt{L_{\text{MSE}}}$ as it is commonly used by the previous research. We split the training dataset into roughly a 90% - 10% fashion for training and validation respectively. The validation set consisted of 3 extracted segments from the training set of 1000 consecutive frames each.

Scores and visualizations: We train for 40 epochs and stop when no improvement takes place on the validation set anymore. The training model saved is the one with the minimum validation loss score of $L_{\text{RMSE}} = 0.0467$. Due to the aggressive dropout that we incorporate, the training loss was highly non-monotonic. We run the model on the test set and record the smoothness and the steering loss scores. Results (Table III) show that the proposed multimodal RNN-LSTM model outperforms community models with a $L_{\text{RMSE}} = 0.0479$. Furthermore, the smoothness (γ) score is 0.761 while the Udacity dataset smoothness score is 0.609. Thereby, we claim that the proposed model output is relatively close in smoothness to the actual driving experience recorded. Figure 7 shows the full prediction and the target values for the full test segment. Notice the high similarity between both ground and regressed results.

[5]github.com/udacity/self-driving-car/tree/master/challenges/challenge-2

Model/Metric	L_{RMSE}	Smoothness (γ)
Target (Benchmark)	-	0.609
Constant 0° steering	0.2070	-
Multimodal RNN-LSTM (HPO)	**0.0479**	0.761
Komanda team[5]	0.0482	-
Du et al. [10]	0.0709	-
Chi et al. [11]	0.0609	-

TABLE III: Community results on the Udacity dataset.

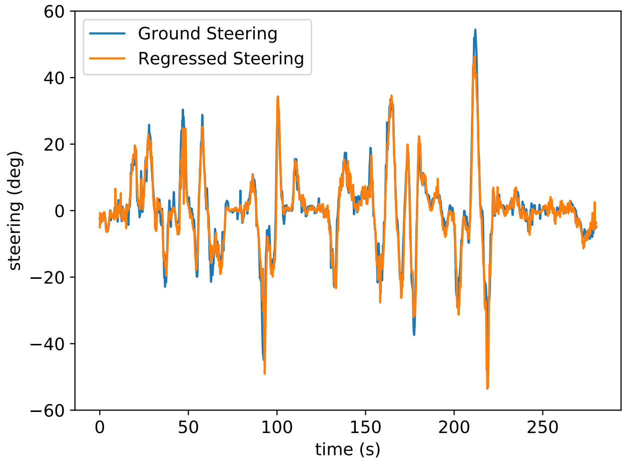


Fig. 7: The target and the prediction steering values on the Udacity test set.

VI. Conclusions and Outlook

We examine empirically the problem of end-to-end learning for autonomous steering using multiple deep learning concepts. We conclude that feeding a multimodal input of spatial and optical flow information benefits the learning process the most. We validate our hypothesis that the RNN-LSTM generates the smoothest output. Through the analysis of the results, we propose a multimodal recurrent model that outperforms state-of-the-art work on the open-source dataset of Udacity. Further, We affirm empirically an off-lane recovery mechanism through the inclusion of correction frames.

Several future research aspects are intriguing. In the same manner of feeding explicitly information about motion, incorporating the smoothness in the cost function could yield an enhancement in the overall model performance. Additionally, we see that a reinforcement learning domain may provide a more generic solution. However, the lack of a fail-safe realistic environment holds the research back. The use of Generative Adversarial Networks (GANS) [28, 29] to regenerate a simulator's images to real images could be one possible solution to provide a fail-safe learning setup.

Acknowledgements

This work was supported by the BrainLinks-BrainTools Cluster of Excellence funded by the German Research Foundation (DFG, grant number EXC 1086).

References

[1] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2722–2730.

[2] H. Zhang, A. Geiger, and R. Urtasun, "Understanding high-level semantics by modeling traffic patterns," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 3056–3063.

[3] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, "3d traffic scene understanding from movable platforms," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 5, pp. 1012–1025, 2014.

[4] S. Chen, S. Zhang, J. Shang, B. Chen, and N. Zheng, "Brain-inspired cognitive model with attention for self-driving cars," *IEEE Transactions on Cognitive and Developmental Systems*, 2017.

[5] T. Hwang, "Computational power and the social impact of artificial intelligence," *arXiv preprint arXiv:1803.08971*, 2018.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[7] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, and U. Muller, "Explaining how a deep neural network trained with end-to-end learning steers a car," *arXiv preprint arXiv:1704.07911*, 2017.

[8] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[9] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in neural information processing systems*, 1989, pp. 305–313.

[10] S. Du, H. Guo, and A. Simpson, "Self-driving car steering angle prediction based on image recognition," *Department of Computer Science, Stanford University, Tech. Rep. CS231-626*, 2017.

[11] L. Chi and Y. Mu, "Deep steering: Learning end-to-end driving model from spatial and temporal visual cues," *arXiv preprint arXiv:1708.03798*, 2017.

[12] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," *arXiv preprint*, 2017.

[13] Z. Yang, Y. Zhang, J. Yu, J. Cai, and J. Luo, "End-to-end multi-modal multi-task vehicle control for self-driving cars with visual perception," *arXiv preprint arXiv:1801.06734*, 2018.

[14] M. Bojarski, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, U. Muller, and K. Zieba, "Visualbackprop: efficient visualization of cnns," *arXiv preprint arXiv:1611.05418*, 2016.

[15] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4694–4702.

[16] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko, "Sequence to sequence-video to text," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4534–4542.

[17] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2625–2634.

[18] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *European conference on computer vision*. Springer, 2004, pp. 25–36.

[19] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Scandinavian conference on Image analysis*. Springer, 2003, pp. 363–370.

[20] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4489–4497.

[21] R. Hou, C. Chen, and M. Shah, "An end-to-end 3d convolutional neural network for action detection and segmentation in videos," *arXiv preprint arXiv:1712.01111*, 2017.

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[23] S. S. Girija, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," 2016.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[25] L. Fridman, D. E. Brown, M. Glazer, W. Angell, S. Dodd, B. Jenik, J. Terwilliger, J. Kindelsberger, L. Ding, S. Seaman, *et al.*, "Mit autonomous vehicle technology study: Large-scale deep learning based analysis of driver behavior and interaction with automation," *arXiv preprint arXiv:1711.06976*, 2017.

[26] S. Falkner, A. Klein, and F. Hutter, "Combining hyperband and bayesian optimization," in *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS), Bayesian Optimization Workshop*, 2017.

[27] H. Hoos and K. Leyton-Brown, "An efficient approach for assessing hyperparameter importance," in *International Conference on Machine Learning*, 2014, pp. 754–762.

[28] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[29] J. Zhang, L. Tai, Y. Xiong, M. Liu, J. Boedecker, and W. Burgard, "Vr goggles for robots: Real-to-sim domain adaptation for visual control," *arXiv preprint arXiv:1802.00265*, 2018.