End-to-End Deep Learning for Steering Autonomous Vehicles Considering Temporal Dependencies

Hesham M. Eraqi*[1], Mohamed N. Moustafa†[1], and Jens Honer‡[2]

[1]Department of Computer Science and Engineering, The American University in Cairo, Egypt
[2]Valeo Schalter und Sensoren GmbH, Germany

Abstract

Steering a car through traffic is a complex task that is difficult to cast into algorithms. Therefore, researchers turn to training artificial neural networks from front-facing camera data stream along with the associated steering angles. Nevertheless, most existing solutions consider only the visual camera frames as input, thus ignoring the temporal relationship between frames. In this work, we propose a Convolutional Long Short-Term Memory Recurrent Neural Network (C-LSTM), that is end-to-end trainable, to learn both visual and dynamic temporal dependencies of driving. Additionally, We introduce posing the steering angle regression problem as classification while imposing a spatial relationship between the output layer neurons. Such method is based on learning a sinusoidal function that encodes steering angles. To train and validate our proposed methods, we used the publicly available Comma.ai dataset. Our solution improved steering root mean square error by 35% over recent methods, and led to a more stable steering by 87%.

1 Introduction

The 2015 Global Status Report of the World Health Organization (WHO) reported an estimated 1.25 million deaths yearly due to road traffic worldwide [16]. With approximately 89% of accidents being due to human errors, autonomous vehicles will play a vital role to significantly reduce this number and ultimately to save human lives. With the ability to shift the task of explicitly formulating rules to designing a system that is able to learn those rules, Artificial Intelligence (AI) will play an important role to realize this vision [7]. Autonomous vehicles will provide greater mobility for old and disabled people and could reduce energy consumption in transportation by as much as 90% [4]. Also, it will translate into less traffic congestion and associated air pollution [4].

Lateral control of a vehicle is one of the most fundamental tasks in the design of algorithms to control autonomous vehicles. A human can achieve this almost solely based on visual clues. In AI terms, the human acts as an end-to-end system, i.e. intermediate representations of a driving lane, a planned trajectory or the relation between hand coordination and steering direction may, and probably do, exist somewhere within the brain of the driver but are never explicitly formulated in terms of interfaces.

To understand and recreate the ability to steer a vehicle based on visual clues within an AI system is thus of fundamental importance. To the authors knowledge, the first work in this regard was done by Pomerleau [18] in 1989 that used a multilayer perceptron (MLP). Since that work, the computational power dramatically increased thanks to massive parallelization in modern GPU's. In combination with modern network architectures and concepts like Convolutional Neural Networks (CNN) [12] that

*heraqi@aucegypt.edu
†m.moustafa@aucegypt.edu
‡jens.honer@valeo.com

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

wasn't known back then. Those advances allowed for having methods that use CNN for end-to-end learning of autonomous vehicles steering [2], [10], [3] that are based on classifying driving scene frame by frame. By design such methods are not equipped to incorporate the temporal relation between image frames and hence cannot learn motion features.

Recurrent Neural Networks (RNN's) [26], [19], [14] represent a class of artificial neural networks that uses memory cells to model the temporal relation between input data and hence learn the underlying dynamics. With the introduction of so called Long Short-Term Memory (LSTM) [9], i.e. the ability to remember selectively, modeling long-term relationships became possible within RNN's.

In this paper we demonstrate quantitatively that a Convolutional Long Short-Term Memory Recurrent Neural Networks (C-LSTM) can significantly improve end-to-end learning performance in autonomous vehicle steering based on camera images. Inspired by the adequacy of CNN in visual feature extraction and the efficiency of Long Short-Term Memory (LSTM) Recurrent Neural Networks in dealing with long-range temporal dependencies our approach allows to model dynamic temporal dependences in the context of steering angle estimation based on camera input.

Posing regression problems as deep classification problems often shows improvements over direct regression training of CNN's [20]. We argue that it can still be further improved. Classification tasks assume independence between the output neurons that encode the different classes. However, this assumption loses validity if the classification is used to model a regression. Because, in such case, classes neurons that are spatially close to each other should infer convergent decisions. Here, we propose a method to introduce correlation between the class neurons and thus bridge the gap between full classification problems and regression problems.

2 System Overview

The system we are investigating is comprised of a front-facing RGB camera and a composite neural network consisting of a CNN and LSTM network that estimate the steering wheel angle based on the camera input. Camera images are processed frame by frame by the CNN. The resulting features are then processed within the LSTM network to learn temporal dependences as detailed in section 3. The steering angle prediction is calculated via the output classification layer after the LSTM layers. For full deployment, the steering angle prediction is transmitted via a steer-by-wire system to appropriate actuators.

In the training phase, this setup is extended to include the ground-truth steering angle input together with a filtering mechanism and an encoder, as well as a Backpropagation algorithm [21], [13] to perform the actual training (see figure 1). The details of the training process are described in section 4. Figure 1 shows a simplified block diagram of the proposed system during training and deployment phases.

Steering wheel sensors are not synchronized with the camera sensor. Typically, the steering wheel angle information is communicated via the vehicle Controller Area Network (CAN) at rates of 100Hz, i.e. at a significantly higher rate than the frame rate per second (FPS) of a camera. We use a low pass

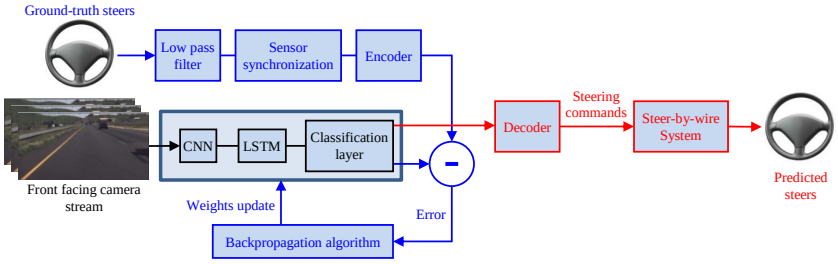

Figure 1: Simplified block diagram of the proposed system. A front-facing RGB camera is used to capture the driving scene and ground-truth steering wheel angles due to a human driver are recorded. During the system deployment phase, a decoder is used to translate the classification layer activations into a steering angle. The blue and red paths represent training and deployment phases respectively.

filter to reduce the steering wheel sensor measurement noise. Then the smoothed signal is sampled to associate a ground-truth steering angle to each single camera frame.

3 C-LSTM Architecture

The proposed Convolutional Long Short-Term Memory Recurrent Neural Network (C-LSTM) architecture combines a deep CNN hierarchical visual feature extractor with a model that can learn to recognize long-term temporal dynamics. Figure 2 depicts the temporal input sequence as processed in our C-LSTM. At each timestamp t the input frame X_t is processed in a CNN based on the C-LSTM architecture.

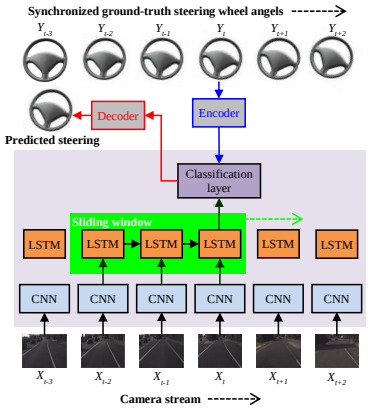

Figure 2: An overview of the proposed C-LSTM architecture unrolled across time. A deep CNN learns to extract best driving scene features, and then the resultant sequential feature vectors are passed into a stack of LSTM layers. The sliding window allows the same frame X_i to be used to train different ground-truth steering angles Y_i, but at different states of the LSTM layers. Classification layer is connected via the blue and red paths during the system training and deployment phases respectively.

CNN's have been recently applied for large scale recognition problems. The training of deeper CNN architectures is proven to be very successful in such problems [12], [24], [22]. This motivated us to choose our CNN to follow the architecture of deepest state-of-the-art CNN's, and we apply the concept of transfer learning [1], [6]. The CNN is pre-trained [1], [6] on the Imagenet dataset [5], [22] that features 1.2 million images of approximately 1000 different classes and allows for recognition of a generic set of features and a variety of objects with a high precision. Then, we transfer the trained neural network from that broad domain to another specific one focusing on driving scene images.

The LSTM then processes a sequence of w fixed-length feature vectors (sliding window) from the CNN as depicted in figure 2. In turn, the LSTM layers learn to recognize temporal dependences leading to a steering decision Y_t based on the inputs from X_{t-w} to X_t. Small values of t lead to faster reactions, but the network learns only short-term dependences and susceptibility for individually misclassified frames increases. Whereas large values of t lead to a smoother behavior, and hence more stable steering predictions, but increase the chance of learning wrong long-term dependences.

In contrast to typical RNN training that is based on fixed subsequent batches, the sliding window concept allows the network to learn to recognize different steering angles from the same frame X_i but at different temporal states of the LSTM layers. Both the CNN and LSTM weights are shared across different steps within the sliding window and in principle this allows for arbitrarily long window size w.

As detailed in section 4, we pose the steering angle regression as a classification problem. This is why the single number representing the steering angle Y_t is encoded to a vector of classification layer neurons' activations. A fully-connected layer with $tanh$ activations is used for the classification layer.

For the domain-specific training, the classification layer of the CNN is re-initialized and trained on camera road data. Training of the LSTM layer is conducted in a many-to-one fashion; the network

learns the steering decisions that are associated with intervals of driving. It is important to set the learning rates of each layer appropriately during learning with the Backpropagation algorithm. The classification layer and the LSTM layers use a larger learning rate because it has been initialized with random values. Note that both the CNN and the LSTM are trained jointly at the same time.

4 Classification Layer

Autonomous vehicles steering prediction is a continuous value regression problem. Given sensor readings, the system predicts the steering angle as a single continuous value. Neural networks, including CNN, can learn regression problems by using a signal neuron at the output classification layer for the regressed steering angle. This direct approach is adopted by the recent works that tackle this problem in literature [2], [10], [3].

In [20], regression formulation through a deep classification followed by expected value refinement has been introduced to improve accuracy of the apparent age estimation from images. The regression problem is posed as a deep classification problem followed by a $softmax$ expected value refinement. We argue that such approach can still be further improved mainly because it has two downsides.

The first downside is that it neglects the topological relation between output classes, i.e. the network has no notion of how different two steering angles are. Figure 3 emphasizes the topological problem. With the steering angles being mapped to discrete output neurons, each output neuron i spans a small steering range of α_i. The figure depicts two prediction example cases that both yield the same loss for conventional loss metrics like Negative Log Likelihood (NLL) or mean squared error (MSE).

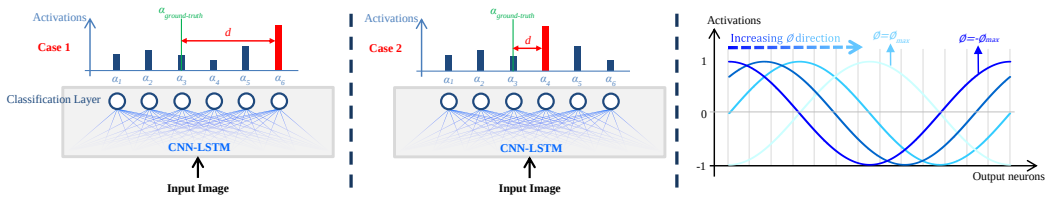

Figure 3: In case 1, low activations occur near the ground-truth neuron, whereas the neuron with peak activation is situated on the far right side, i.e. far away from the ground truth. In contrast, case 2 features the neuron with peak activation close to the neuron that is expected by the ground-truth and thus a better prediction. Yet loss metrics that neglect the spatial relations will yield the same penalties in both cases. The figure also shows Sine waves encoding at different steering angles. The dotted arrow denotes the increasing ϕ direction.

The second drawback of the conventional formulation of regression as a classification problem is that the amount of required training data scales with the number of discrete angles, i.e. classes. The expanding number of class labels requires collecting more training data such that all class labels contain sufficient training volumes. Conversely, if the training data is limited, so is the number of classes.

To solve this problem we introduce a spatial relationship between the classification layer neurons. The method is based on learning an arbitrary function that encodes the steering angle. It's inspired from our patent regarding pose angle estimation of an object in an input image [15]. The steering wheel sensor provides a steering angle ϕ ranging from $-\phi^{\circ}_{max}$ (extreme turning to the left) to ϕ°_{max} (extreme right turn). $\phi = 0^{\circ}$ means the vehicle driving straight forward. We choose the encoding function to be a sine wave and the steering angle to be its phase shift as in (1):

$$Y_i = \sin\left(\frac{2\pi(i-1)}{N-1} - \frac{\phi\pi}{2\phi_{max}}\right), \quad 1 \leq i \leq N, \tag{1}$$

where Y_i is the activation of the output neuron i and N is the number of output layer neurons, i.e. classes. Such choice for the learned function and its parameter(s) encoding steering, to be a sine wave and its phase shift, guarantees gradual changes in steering to cause gradual changes of the output layer activations. Figure 3 shows example sine waves encoding different steering wheel angles.

Figure 4 shows how the proposed method is used for learning efficient vehicle steering. $Tanh$ activation functions are used for the classification layer neurons to allow it shaping sine waves with

amplitude from -1 to 1. During training, the ground-truth steering angle ϕ is encoded as a sine wave function as in (1). On the other hand, the classification layer output uses a Least Squares regression in order to fit the predicted function. The Backpropagation loss function is calculated as the root mean square error (RMSE) between the two waveforms over batches of sequential data.

During deployment the steering angle is decoded from the Least Squares regression result. The decoder calculates the sine wave phase shift that contains the information about the steering angle. This new concept of network output formulation as a fitted sinusoidal function can be extended to other application domains where regression is posed as a classification problem for better performance; like Face Age Detection, Face Pose Estimation, House Price Estimation, etc.

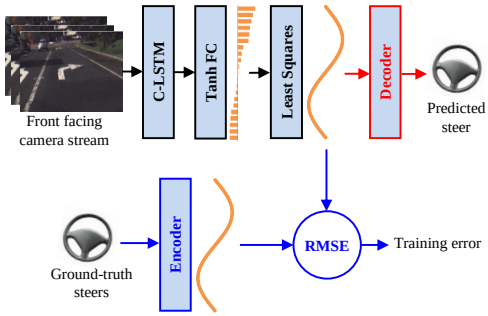

Figure 4: Fitting a sin wave that encodes driving steering wheel angle. The blue and red paths represent training and deployment phases respectively.

5 Experimental Results

In this section, we first introduce the dataset we used for experimental work and the system evaluation metrics. Then we present the implementation details of our method, describe experimental setups, and discuss results.

5.1 Dataset and Evaluation Metrics

We trained and validated our system on the recently publicly released database by "comma.ai" in [23]. It contains 7.26 hours of highway and city driving data that are divided in 11 videos captured during both day and night. The dataset also has several sensors that were measured in different frequencies. In this work, we only use the camera frames and steering wheel angle signal. In [2], the authors trained their system using about 72 hours of driving data that is not publicly available. We aim to provide a baseline performance for future works to benchmark against.

We choose the first and third video files, having 1 hour of driving, to form the testing set, and the remaining nine files were kept for training and validation. That balances the training and testing sets between day and night driving. We sampled both of the training and testing datasets to 2 Hz. In [27], Comma.ai dataset is used to validate Deep Predictive coding networks. In that work, random 5% chunks are sampled from each video file in the dataset and kept for validation and testing. Such split strategy causes the correlation between the training and testing data to be too high, and the steering prediction becomes a much easier problem. We separate complete video files for testing such that testing data contains driving sessions that are completely different from those of the training data, i.e.; different locations, time of day, and driving conditions.

Both Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) can express average system prediction error. For steering angle prediction problem, large errors are particularly undesirable. We choose to report our results using RMSE, defined as in (2) to gives a relatively high weight to large errors.

$$RMSE = \sqrt{\frac{1}{|D|}\sum_{i=1}^{|D|}(G_i - P_i)^2}, \tag{2}$$

where G_i and P_i are the ground-truth and predicted steering angles for a frame i respectively in a testing set having a total of $|D|$ frames. Both angles are measured in angular degrees.

MAE and RMSE can range from 0 to $\inf$ and are indifferent to the direction of errors. They cannot infer how much stable steering is. Thus, we use another metric W that measures the whiteness of the predicted steering signal as in (3). The smaller the value of W, the more smooth steering variations are predicted.

$$W = \frac{1}{|D|} \sum_{i=1}^{|D|} \left(\frac{\partial P(t)}{\partial t} \bigg|_{t=i} \right)^2 \tag{3}$$

5.2 Results

We report in this section our experimental results on the testing dataset we defined earlier. In table 1, we report performance of using a CNN in learning steering using regression; the CNN has a single output neuron. Firstly, we use a simple CNN that follows the architecture introduced in [2]. And then we apply transfer learning using deeper state of the art networks, which are the third version of Inception network [25] and Resnet network having 152 layers [8].

Table 1: Regression using CNN

CNN Network	Performance	
	RMSE [Degrees]	Whiteness [Degrees / Time unit]
Simple CNN [2]	23.30	65.8
Inception V3	18.67	43.9
Resnet 152	17.77	39.1

Table 1 confirms that deeper networks perform better. We conducted cross-validation using different optimizers, network regulation, and learning rates until we found best choices. The simple CNN and the fully connected layers had a learning rate of 10^{-3}, while the convolutional layers had a learning rate of 10^{-5} for transfer learning . Adam optimization [11] was used to train all networks. Applying batch normalization and dropout techniques helped minimizing over-fitting.

Table 2 compares the direct regression performance with the state-of-the-art classification method in [20]. Subsequently, it compares them with our proposed classification method; by fitting a sinusoidal function. Most importantly, it reports the results of our complete solution which uses the C-LSTM architecture. The output feature vector from the CNN is of length 2048. Empirically, θ, ϕ_{max}, and σ^2 are chosen to be equal to $4°$, $190°$, and 80 respectively. Hence, the number of output neurons N is equal to 95 neurons. We found that a sliding window stride of 1 time step resulted into most accurate predictions. Empirically, we used 2 LSTM layers, with each layer having 500 neurons and w covering 5 seconds of driving.

Table 2: Comparison of CNN And C-LSTM For Regression and Classification

CNN Network	CNN Performance		C-LSTM Performance	
	RMSE [Degrees]	Whiteness [Degrees / Time unit]	RMSE [Degrees]	Whiteness [Degrees / Time unit]
Regression	17.77	39.1	16.01	9.7
Classification,using NLL [20]	18.70	54.1	17.84	10.0
Classification by,sine wave fitting	17.44	43.9	**14.93**	**8.2**

Table 2 demonstrates that our proposed method for classification performed better than both of the conventional classification method and the direct regression of steering angle method. Most importantly, it shows that using our C-LSTM architecture led to a significant improvement of steering angle prediction in terms of accuracy and stability reflected by the predicted steering signal RMSE and whiteness respectively. The signal was more accurate than the state-of-the-art solution by 35% and was more stable by 87%. A steering wheel angle RMSE slightly less than $20°$ is still an achievement, since modern car "steering ratio" is in the range $12:1$ to $20:1$ [17]. Consequently, a steering wheel RMSE of $20°$ is actually equivalent to approximately only $1°$ of vehicle wheels (tyres) off-direction.

6 Conclusions

We propose and benchmark a C-LSTM architecture that allows learning both visual and temporal dependencies of driving. We also introduce posing the steering angle regression problem as a deep classification problem by imposing a spatial relationship between the output layer neurons. That concept of output formulation as a fitted sinusoidal function can be extended to other application domains where regression is posed as a classification problem for better performance. Our solution achieved 35% steering RMSE improvement and led to a more stable steering by 87% compared to recent methods tested on the benchmark Comma.ai publicly available dataset. The future work should focus on verifying the proposed method via simulation and analyzing the effect of the number of output neurons on the steering performance.

References

[1] Yoshua Bengio et al. Deep learning of representations for unsupervised and transfer learning. *ICML Unsupervised and Transfer Learning*, 27:17–36, 2012.

[2] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[3] Mariusz Bojarski, Philip Yeres, Anna Choromanska, Krzysztof Choromanski, Bernhard Firner, Lawrence Jackel, and Urs Muller. Explaining how a deep neural network trained with end-to-end learning steers a car. *arXiv preprint arXiv:1704.07911*, 2017.

[4] Austin Brown, Brittany Repac, and Jeff Gonder. Autonomous vehicles have a wide range of possible energy impacts. Technical report, NREL, University of Maryland, 2013.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[6] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *Icml*, volume 32, pages 647–655, 2014.

[7] Hesham Eraqi, Youssef EmadEldin, and Mohamed Moustafa. Reactive collision avoidance using evolutionary neural networks. In *Proceedings of the 8th International Joint Conference on Computational Intelligence*, volume 1, 2016.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[10] Joel Janai, Fatma Güney, Aseem Behl, and Andreas Geiger. Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art. *arXiv preprint arXiv:1704.05519*, 2017.

[11] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[13] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[14] Haobo Lyu, Hui Lu, and Lichao Mou. Learning a transferable change rule from a recurrent neural network for land cover change detection. *Remote Sensing*, 8(6):506, 2016.

[15] Mohamed Nabil Moustafa. System and method for pose-angle estimation, October 25 2005. US Patent 6,959,109.

[16] World Health Organization. *Global status report on alcohol and health 2014*. World Health Organization, 2014.

[17] Hans Pacejka. *Tire and vehicle dynamics*. Elsevier, 2005.

[18] Dean A Pomerleau. Alvinn, an autonomous land vehicle in a neural network. Technical report, Carnegie Mellon University, Computer Science Department, 1989.

[19] Paul Rodriguez, Janet Wiles, and Jeffrey L Elman. A recurrent neural network that learns to count. *Connection Science*, 11(1):5–40, 1999.

[20] Rasmus Rothe, Radu Timofte, and Luc Van Gool. Dex: Deep expectation of apparent age from a single image. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 10–15, 2015.

[21] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[22] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[23] Eder Santana and George Hotz. Learning a driving simulator. *arXiv preprint arXiv:1608.01230*, 2016.

[24] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[25] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[26] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[27] Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. End-to-end learning of driving models from large-scale video datasets. *arXiv preprint arXiv:1612.01079*, 2016.