Two-stream Convolutional Networks for End-to-end Learning of Self-driving Cars

Nelson Fernandez[a,†]
[a] Renault Automotive
nelson.fernandez-pinto@renault.com
[†] Previously at Axionable Labs, Paris France

Abstract

We propose a methodology to extend the concept of Two-Stream Convolutional Networks to perform end-to-end learning for self-driving cars with temporal cues. The system has the ability to learn spatiotemporal features by simultaneously mapping raw images and pre-calculated optical flows directly to steering commands. Although optical flows encode temporal-rich information, we found that 2D-CNNs are prone to capturing features only as spatial representations. We show how the use of Multitask Learning favors the learning of temporal features via inductive transfer from a shared spatiotemporal representation. Preliminary results demonstrate a competitive improvement of 30% in prediction accuracy and stability compared to widely used regression methods trained on the Comma.ai dataset.

1 Introduction

Decision making in the spatiotemporal domain is a key issue for autonomous driving systems [1]. The current paradigm is the implementation of a Convolutional Neural Network (CNN) to perform direct steering commands regression from raw images [2]. A forward-facing camera records video footage of the road, which is tagged with the vehicle's cinematic measurements, such as speed, wheel angle and acceleration. The data is then used to train the CNN regressor with a supervised learning approach. This system has proven to be efficient on extracting spatial cues [3] and achieves respectable accuracy. The main limitation is that frames are analyzed individually without taking into account previous actions. Therefore, any temporal information is lost due to this abstraction.

Following its introduction by Symonyan and Zisserman (2014) [4], Two-Stream Convolutional Networks have become widely adopted as the preferred method for action recognition in videos. The main advantage is the exploitation of the temporal dynamics captured by the optical flow between adjacent frames. In this case, the temporal stream is prone to decoding features instead of making an estimation of the motion field. To the best of our knowledge, the use of this architecture in regression tasks, especially autonomous steering, has not been studied in depth.

It is widely believed that the use of temporal features together with spatial characteristics could lead to better model predictions. Currently, the main research efforts are focused on the use of Recurrent Neural Networks (RNN) [5] and 3D-Convolutional Neural Networks (3D-CNN) [6]. In contrast to intuition, the use of 2D-CNNs could bring some advantages compared to RNNs, in particular, a faster and more straightforward training. Also, 2D-CNNs are less likely to overfit when performing temporal tasks [7]. On the other hand, 3D-CNNs must perform an implicit estimation of the optical flow while trying to learn temporal features through the depth of the input image tensor, which makes training difficult.

†This work has been supported by Axionable Labs, Paris 75003, France.

NeurIPS 2018 Workshop on modeling and decision-making in the spatiotemporal domain, Montréal, Canada.

The principal motivation of this work is to demonstrate that Two-Stream Convolutional Networks are a robust alternative to perform end-to-end learning with temporal cues. This article describes the preliminary results tested on Comma.ai's dataset [8] compared to other regression methods.

2 Methods

Based on the Two-Stream Convolutional Network architecture, it is possible to design a custom self-driving system that takes advantage of spatial and temporal features using 2D-CNNs. This system is composed of two visual streams taken from a single front-facing camera and its optical flow calculated from the previous image. The optical flow can be estimated efficiently in almost real time using the current accelerated computing technology. In addition, specialized hardware can be integrated to obtain live optical flow streams. A recording device uses the controller area network (CAN) bus to collect steering wheel commands directly from the vehicle's computer. This data is used to train a regressor, which provides a prediction of the required steering angle based on inputs. The network output is decoded and sent to a proportional–integral–derivative controller (PID), to finally steer the position of the wheels. Figure 1 shows an overview of the proposed self-driving car system.

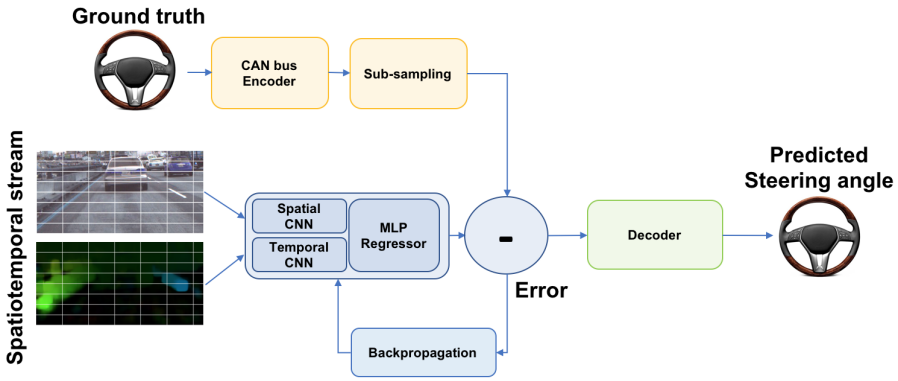

Figure 1: Proposed Two-Stream self driving navigation system.

Figure 2 shows the proposed network architecture that incorporates key elements of Two-Stream Convolutional Networks and 2D-CNNs to steering angle regression. The ensemble is composed by two identical CNN branches similar to [9]. Global spatial pooling is applied in the last convolutional layers, aggregating information from all dimensions into single real values. This global aggregation considerably reduces the number of parameters and decreases the risk of overfitting [10]. The resulting embeddings are merged with a fusion layer of type element-wise multiplication, forming a shared spatiotemporal representation. A multi-layered perceptron (MLP) with linear output neurons performs the regression. Following the Two-Stream Networks logic, the first branch is intended to learn spatial features extracted from raw images [3]. The second learns temporal characteristics from the estimated relative displacement of objects between frames.

Based on the inputs, it is possible to train the network to simultaneously predict the current and previous state of the target variable using Multitask Learning (MTL). As the goal of architecture is to capture short-term temporal dependencies, we assume that on the horizon studied, the inputs provide the information needed to perform the regression. The main motivation of performing MTL is to favor the learning of temporal features via inductive transfer [11]. By consequence, the auxiliary target is dropped during the inference step. The architecture has about 45 million parameters and is trained using backpropagation.

The model was trained with 6.2 driving hours from the Comma.ai dataset [8], corresponding to approximately 220K image/optical flow pairs of size 160x320x3 sampled at 10 Hz. The optical flow was calculated using the Farnebäck's [12] implementation available in OpenCV. The estimated motion field was mapped to a standardized RGB color wheel. Dropout and data augmentation strategies, including spatial and photometric transformations were applied to avoid overfitting. Due to the small size of the training set, this work is more aimed at obtaining a benchmark rather than training an autopilot ready for production.

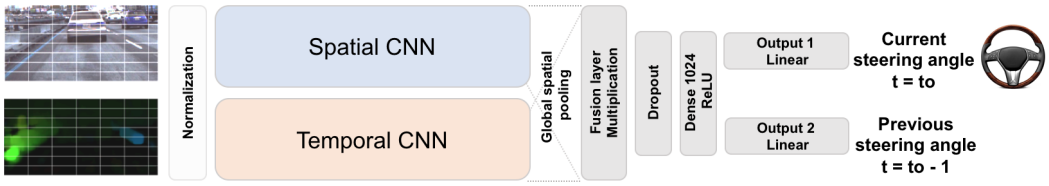

Figure 2: Proposed Two-Stream Convolutional Network architecture.

A two-step training strategy was used, with 60% of the training data for feature learning, and 40% for fine-tuning. During the first step, the CNN branches are trained independently, the weights obtained are used as initialization of the second stage. In the second step, we fine-tune the architecture by focusing on the MLP. We set the following hyper parameters: Loss function of type Mean Squared Error (MSE), Adam optimizer [13] with learning rate 1e-4 for feature learning and 0.5e-4 for fine tuning, and a batch size of 64 for feature learning, and 8 for fine tuning. The model trained for 30 epochs in the feature learning step and for 1 epoch for fine tuning. The architecture was implemented using the Keras functional API running on a machine with a single NVIDIA M6 GPU and 120Gb of RAM.

The test set comprised 56 minutes of video taken from the second and third files of the Comma.ai dataset, approximately 33K images sampled at 10Hz. This was done to prevent data leakage due to similar frames in the training and test partitions in accordance with [5]. As no examples of these folders were included in the training set, the regression task becomes harder, measuring the generalization capabilities of the model.

The main evaluation metric was the Root Mean Squared Error (RMSE), that measured the accuracy of the steering angle regression. We also used the whiteness [5], that measured the smoothness of the predictions over time. This quantity is based on the square of the first time derivative calculated according to expression (1). The whiteness reflects the temporal dynamics of the steering task. Low values mean smooth steering, and therefore a more comfortable driving experience for users.

$$Whiteness = \sqrt{\frac{1}{D}\sum_{i=1}^{D}\frac{\partial P_t}{\partial t}^2}\,[\frac{degrees}{time}] \tag{1}$$

3 Results

The first experiment is intended to evaluate the effect of using optical flows instead of raw images to train single 2D-CNNs. For this, we trained two widely used architectures to perform steering angle regression directly from images and optical flows respectively. The main idea is to assess the capability of CNNs to learn temporal features from the displacement of objects between consecutive frames. The results of this experience are shown in table 1.

Table 1: Test set RMSE and whiteness of single-CNN architectures.

	Trained with raw images		Trained with optical flows	
Model	RMSE[degrees]	Whiteness	RMSE [degrees]	Whiteness
Simple-CNN [2]	20.55	8.84	**14.21**	8.74
Inception V3	17.76	7.05	**12.58**	7.15

The above results show that whichever CNN architecture is utilized, the optical flow produces significant accuracy gains in the test set compared to raw images. We also see some improvement associated to the architecture sophistication of Inception V3 [9] compared to simple CNNs. These results suggest that the object displacement representation is convenient to learn specific features improving generalization. The fact that the whiteness did not improve, implies that the networks are learning features only as spatial representations.

In the second experiment, we evaluate the proposed architecture alongside other widely-used steering angle regression methods. Table 2 shows that Two-Stream Convolutional Network outperformed state-of-the-art 2D-CNN models in prediction accuracy and stability, with a whiteness closer to the human driver.

Table 2: Test set steering wheel angle regression error.

Architecture	RMSE [degrees]	Whiteness [degrees/time]
Human driver	N/A	4.36
Comma.ai 2D-CNN	23.99	9.81
Nvidia PilotNet	20.55	9.23
Inception V3	17.76	7.05
Two-Stream Network	**12.52**	**4.97**

Figure 3 shows the scatter plot of the test set instances in paired coordinates of instantaneous whiteness (x-axis) and steering angle (y-axis), along with predictions of three different CNN regression models. As the whiteness is proportional to the square of the first time derivative, the figure is a spatiotemporal representation of the steering wheel angle task. The proposed architecture (right) helps to resolve the bias in the spatial and temporal axis of individual CNN streams trained with raw images (left) and optical flows (center). This improvement is achieved by the incorporation of a shared spatiotemporal representation and MTL from a time-related task. The learning of the previous steering angle as an auxiliary task, produces bias correction in the temporal domain, reducing the whiteness. This filtering effect would be difficult to achieve using only single-target individual CNNs.

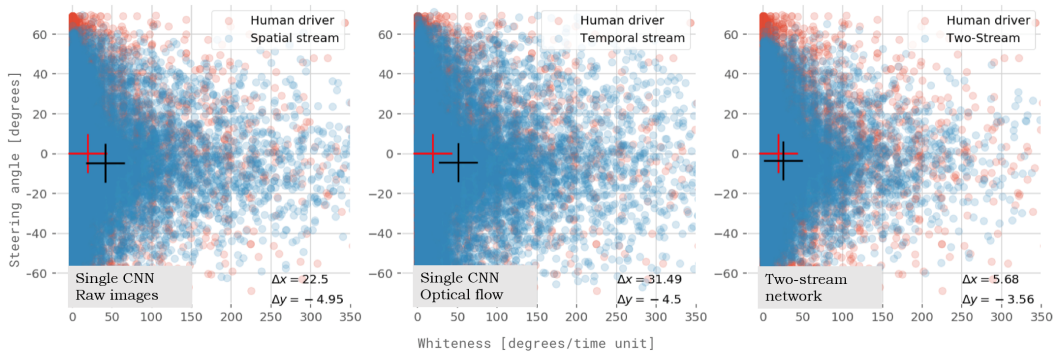

Figure 3: Test set model bias of a spatial stream (left), temporal stream (center) and the proposed architecture (right).

4 Discussion

The main contribution of this work is to demonstrate that Two-Stream Convolutional Networks are a robust alternative to augment end-to-end learning with spatiotemporal cues. We found that the use of optical flows instead of raw images can significantly increase the accuracy of current self-driving systems. The main reason for this is that optical flows provide a convenient representation that can be easily learned by CNNs. However, the use of optical flows alone is not sufficient to incorporate temporal dependencies, as CNNs are biased to learning characteristics as spatial representations.

The proposed Two-Stream architecture learns how to effectively combine spatial and temporal information from a shared spatiotemporal representation that encodes the relative movement of objects between two consecutive frames. The introduction of a related auxiliary task (the previous steering angle) guides the network to discover short-term temporal dependencies via inductive transfer. Then, an important bias correction is achieved in the spatial and temporal domains improving generalization. The results show a competitive performance increase in prediction accuracy and stability of 30%, when compared to widely-used regression methods trained on the Comma.ai data set. The next steps of this work will focus on the study of MTL for autonomous driving regression tasks.

References

[1] Schwarting, W., Alonso-Mora, J., & Rus, D. (2018). Planning and Decision-Making for Autonomous Vehicles. Annual Review of Control, Robotics, and Autonomous Systems, 1, 187-210.

[2] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., ... & Zhang, X. (2016). End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316.

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

[4] Simonyan, K., & Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. In Advances in neural information processing systems (pp. 568-576).

[5] Eraqi, H. M., Moustafa, M. N., & Honer, J. (2017). End-to-end deep learning for steering autonomous vehicles considering temporal dependencies. arXiv preprint arXiv:1710.03804.

[6] Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M. (2015). Learning spatiotemporal features with 3d convolutional networks. In Proceedings of the IEEE international conference on computer vision (pp. 4489-4497).

[7] Xie, S., Sun, C., Huang, J., Tu, Z., & Murphy, K. (2018, September). Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 305-321).

[8] Santana, E., & Hotz, G. (2016). Learning a driving simulator. arXiv preprint arXiv:1608.01230.

[9] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).

[10] Lin, M., Chen, Q., & Yan, S. (2013). Network in network. arXiv preprint arXiv:1312.4400. ISO 690

[11] Caruana, R. (1997). Multitask learning. Machine learning, 28(1), 41-75.

[12] Farnebäck, G. (2003). Two-frame motion estimation based on polynomial expansion. In Scandinavian conference on Image analysis (pp. 363-370). Springer, Berlin, Heidelberg.

[13] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.