Deep Learning for Computer Vision: Final Project Proposal
End to End Learning for Visual Navigation

Ute Schiehlen
ute.schiehlen@tum.de

Natalie Reppekus
n.reppekus@tum.de

Areeb Kamran
areeb.kamran@tum.de

Raymond Chua
raymond.chua@tum.de

Proposal I

1. Introduction

An important component in autonomous vehicles is visual navigation. In this project, we intend to solve this problem using an end to end approach. We propose modeling a video of the front camera of a car as an ordered sequence of frames, with additional motion information in the form of optical flow images computed over adjacent frames to get the steering angle. In addition, to further exploit temporal information, we consider a recurrent neural network that uses *Long-Short term memory* (LSTM) cells as our main architecture.

2. Related Work

End to end learning using *fully convolutional networks* (FCNs) has proven to be successful in many computer vision tasks such as Recognition[8], Object Detection[8], Localization[8, 7] and Semantic Segmentation [9]. One reason why end to end learning is so powerful is that it allows the network to learn the internal representation of the data using the provided training information.

In addition, videos are composed with spatial and temporal components. We consider using dense optical flow information from the image sequences as additional data, which has shown to improve results for action recognition[10] and video classification[6]. These approaches use a two-stream network, one for spatial information and one for motion. The two streams are then merged to compute a fusion score.

Bojarski et al.[3] has shown that end to end learning can also be applied for self-driving cars. They propose a model which consists of one normalization layer, five convolutional layers and three fully-connected layers [3], using only human steering angle as the training signal. The network is able to drive the car autonomously for 90% and 98% of the time during the simulation and on-road tests respectively [3].

3. Our Approach

Although the model from Bojarski et al.[3] has achieved successful results, it did not fully exploit the temporal information of the image sequences. The network is trained with each sample being a single frame from the video. Since the frames are highly dependent, we believe that results can be greatly improved if this is taken into consideration by a network. As such, we consider a model that is capable of learning long-term dependencies. One such network architecture is *Long Short Term Memory* (LSTM)[5], which is a variant of *Recurrent Neural Network* (RNN).

We would like to compare the network described in[3] with a network consisting of a few convolutional blocks based on *Residual Networks*[4] (ResNet), whose output is fed into the LSTM network. If this yields promising results we will extend this architecture by adding another convolutional network which takes optical flow data generated from the image sequences as input. Its output will be concatenated with the output of the convolutional network from before and used as input for the LSTM network.

4. Dataset

We consider using the open dataset provided by Udacity[2], more specifically the CH2_002 training dataset from the second challenge. It contains six different recordings of driving scenarios with a total of approx. 28 minutes. The first one, for example, is recorded in direct sunlight with many lightning changes on a divided highway including turns in the beginning and a lane merge at the end and has a duration of 221 seconds.

We will split this dataset into train, validation and test set with a 60%, 20%, 20% distribution respectively. The JPG images will be used as the input source to our network and the steering wheel angel as the corresponding labels. As stated in the README of this project, the data is a cleaned version complying with the defined naming schema and is supposed to be error free.

5. Methodology

5.1. Network Architecture

Due to the nature of the problem, there might be a high number of frames where the car is just going straight, which results in minimal steering angles. In order to have a uniformly distributed dataset, we split the dataset into multiple bins, based on the steering angles. We can then consider a subset of the training data, such that all the bins are equally represented.

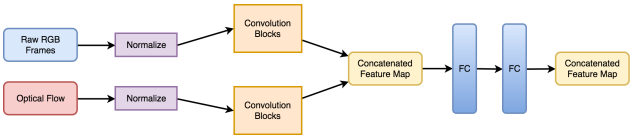

Figure 1. Convolutional Network for our Image Sequences and Optical Flow. Best view in color.

Using this new dataset, for our final network we can consider each training sample as a pair of a RGB image and its corresponding optical flow data for each time step. The training sample is normalized separately. We can then feed the normalized data into our convolutional network for feature extraction which results in a concatenated feature map (Figure. 1).

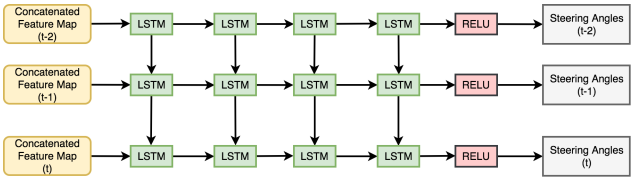

Figure 2. Long-Short Term Memory Network. Inputs are the concatenated feature maps generated for each time step. A rectified linear (RELU) activation layer is applied at the end. Best view in color.

We feed the concatenated feature map into four stacked LSTM layers. A rectified linear activation layer is added to predict the steering angles for each time step. (Figure. 2).

We consider $\boldsymbol{\theta} \in \mathbb{R}^d$ as the set of our network parameters, $\mathbf{x} = (x_1, ..., x_t)$ as our RGB image sequence input and $\mathbf{o} = (o_1, ..., o_t)$ as our Optical Flow sequence input. By denoting f as our network and y as the ground truth steering angle per frame, we define our loss function based on the **Mean Square Error** metric (MSE) :

$$Losst = \frac{1}{2}\Big(f(\hat{y}; \mathbf{x}, \mathbf{o}, \boldsymbol{\theta}) - y\Big)^2 + \frac{\lambda}{2} \mid\mid \boldsymbol{\theta} \mid\mid^2 \qquad (1)$$

where $\hat{y}$ is our predicted steering angle and λ is our regularization parameter.

5.2. Transfer learning and training from scratch

As described in methodology, the first two parts of our approach are based on transfer learning. In the first part we build the network based on[3]. In the second part, we use our trained model from the first part and train the the LSTM layers on top. For the optical flow integration, we only use the pre-trained convolutional layers and train the new convolutional and LSTM layers from scratch.

5.3. Resource management

For development our personal or chair machines are used and the network is trained via AWS services.

6. Outcome

We hope to to train the network successfully. By using Bojarski et al.[3] model as the baseline, the project can be considered a success if there are any improvements to the prediction results.

Proposal II

1. Introduction

In the project we propose to predict the number of crimes in a city. This is a phenomenon which depends on both space and time. However training neuronal network architecture directly on this data would not be feasible. So from the crime data, we propose to build point cloud maps for the city of San Francisco. Each point map can be of a range between just one hour and one day. That way we can generate images for about 10 years from the data available.

2. Our Approach

We are proposing to first generate point clouds or dot map images from data and then train two networks on it. The data itself is available from 2003 to 2017 and can be divided into training data which can be first 12 years and test data which would be the next 3 years.

3. Methodology

Our methodology is to use two networks and use majority voting for final prediction. One network would deal with a temporal information (time series only) and would be an LSTM network. The other network would work with spatial data and information (coordinates) and so would be a convolutional network. We then would combine the results from both for predicting when and where might a crime be expected in a city.

4. Dataset

We would use the open Data set from City of San Francisco's open data portal [1]. It has detailed crime data which can be broken down into multiple categories as well (robbery, manslaughter, auto theft, etc)

References

[1] Map of Police Department Incidents dataset. https://data.sfgov.org/Public-Safety/Map-of-Police-Department-Incidents/gxxq-x39z/data. Accessed: 2017-06-27. 2

[2] Udacity Self-driving-car dataset. https://github.com/udacity/self-driving-car/tree/master/datasets/CH2. Accessed: 2017-06-26. 1

[3] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016. 1, 2

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 1

[5] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. 1

[6] J. Y. Ng, M. J. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 4694–4702, 2015. 1

[7] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Is object localization for free? - weakly-supervised learning with convolutional neural networks. In *CVPR*, pages 685–694. IEEE Computer Society, 2015. 1

[8] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013. 1

[9] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):640–651, 2017. 1

[10] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 568–576, 2014. 1