Comma Speed Challenge: An OpenPilot Approach

Garrett A. Partenza
Northeastern University
440 Huntington Ave, Boston, MA 02115
`partenza.g@northeastern.edu`

Abstract

The contribution of this work is an attempt of the 2017 Comma Speed Challenge. The goal of the challenge is to predict the speed of a car from the view of its own dash camera. In this work, I demonstrate how to integrate recurrent all-pairs field transforms for optical flow alongside an OpenPilot-style network architecture to train. Experimental results indicate this approach performs fairly well. More information regarding the challenge can be found in the CommaAI GitHub repository `https://github.com/commaai/speedchallenge`.

1 Dataset

CommaAI provides a dataset within the challenge repository that consists of 20400 frames from a dashcam video shot at 20 fps, as well as a text file labeling the speed of the car at each frame. The training video is highly diverse, including highway driving, street driving, as well as full stops at stop signs. They also provide an unlabeled test video which you can submit to them through email for blind grading. Both videos are 3 channels and 480 by 640 pixels. The images were not resized, re-sampled, or normalized before training.

2 Methodology

Being that we are trying to detect the intensity of motion within an image, it makes sense to utilize optical flow as a feature in our training pipeline. In short, optical flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and a scene. A better definition in the context of this project is that optical flow is the distribution of apparent velocities of movement of brightness pattern in an image [1]. Algorithmically this can be solved using differentiation between two images and approximations of corresponding pixels, however, this method results in two unknowns and thus is unsolvable without additional constraints. While there are numerous mathematical approaches to fix this issue, recent approaches have found success in using deep learning with the most popular project known as Recurrent All-Pairs Field Transforms (RAFT) Teed and Deng [2] .

Since RAFT is open-sourced [3], I downloaded the pretrained model and converted all video frames into their corresponding optical-flow representation before being inputted into any downstream neural architecture. In this manner, we maintain the edges and shapes within images and use the RGB channels to pass optical flow information.

For the learnable layers, I use a neural architecture style based off of OpenPilot, a shippable level-two self-driving system [4]. The developers of OpenPilot, inspired by value prediction networks in MuZero, use a three level architecture consisting of a feature extraction layer, dynamics layer, and

Not submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021).

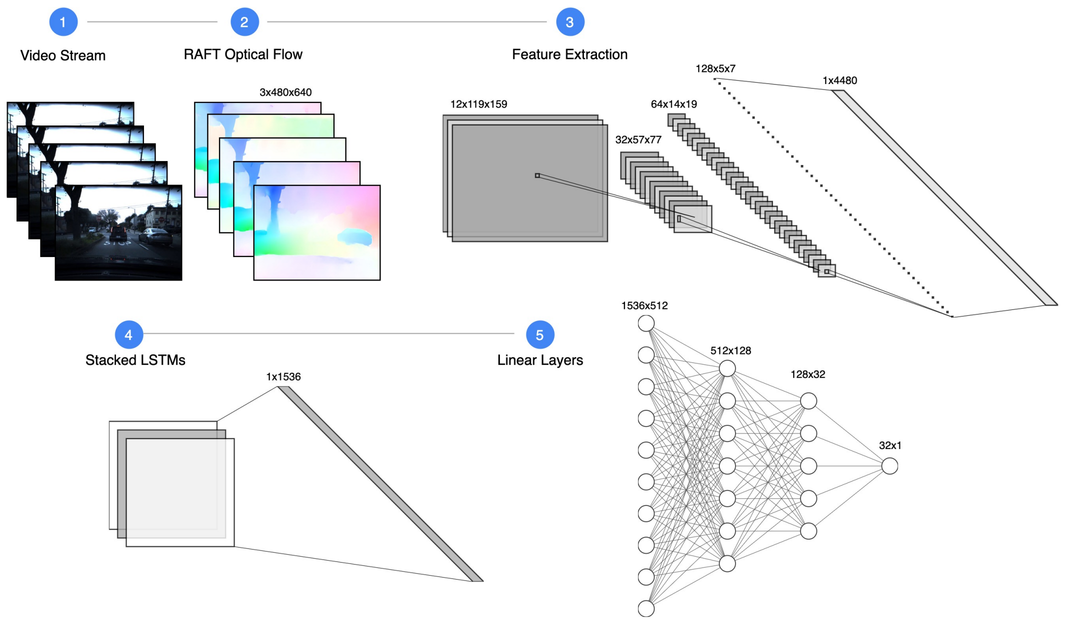

Figure 1: Neural architecture based on CommaAI's OpenPilot

prediction layer. More specifically, they use an efficientnet-b2 for feature extraction, bidirectional-gated recurrent unit for dynamics, and a few dense layers for prediction.

The architecture I personally used for training consists of four stacked convolutional layers for feature extraction of the optical flow images from RAFT. Batch normalization and max-pooling are also applied after each convolutional layer. The output of the convolutional layers is then fed into three stacked bidirectional LSTMs to extract the temporal features. Finally, the last hidden states of each cell are flattened before being fed into four fully connected layers for prediction. Before each dense layer, a dropout of 0.2 was applied to aid in generalization [5]. To meet the requirements of Machine Learning 6140 I train two different networks, one with a recurrent layer and one without, to demonstrate the utility of temporal dimensions in computer-vision for driving tasks.

3 Experimental Setup

For both models, 10,000 test cases were randomly sampled from the challenge dataset with a validation set size of 0.1. Both models were trained for 100 epochs with a batch size of 32 using an Adam optimizer. Learning rate was set to 0.02 alongside an exponential decay scheduler with gamma set to 0.99 and stepped every epoch. For the model which used a recurrent layer, a sequence of length 5 was used which amounts to 0.25 seconds of video. This is a regression task, so predictions of both models were evaluated using the mean squared error between model output and actual speed in MPH. The MSE of the training set and the validation set was logged on every epoch. GPU training was utilized on the Northeastern Discovery compute cluster with a NVIDIA Telsa P100 12GB. Average time to completion of training was around three hours for both models.

4 Results

Figure 2 and Table 1 showcase my experimental results. Note that the y-axis in Figure 2 has been scaled logarithmically. The recurrent model achieves a minimum validation MSE of 4.623 (within 2 MPH) while the non-recurrent model achieves 6.844 (within 2.6 MPH). The Comma Speed Challenge repository states that a mean squared error less than 10 is good, while less than 5 is better and less than 3 is heart (<3, get it?). By this grading scale I conclude that my model achieved above average performance on the validation set.

Experimental Results				
Epoch	**Recurrent Train MSE**	**Recurrent Val MSE**	**Non-Recurrent Train MSE**	**Non-Recurrent Val MSE**
1	37.895	**20.203**	253.05	44.966
3	24.725	**14.506**	25.133	33.43
5	20.636	**11.678**	20.994	34.799
10	16.983	**13.681**	16.755	19.366
25	10.804	**8.664**	10.786	25.176
50	8.133	**5.94**	9.937	9.81
100	5.818	**5.663**	6.891	12.158

Table 1: MSE of both networks for train and validation sets

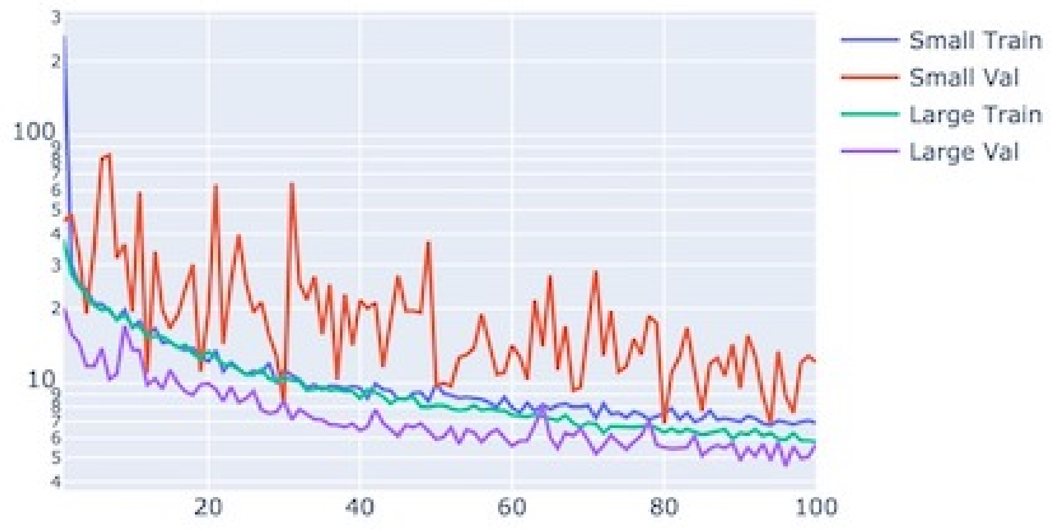

Figure 2: MSE of both networks for train and validation sets

5 Conclusion

This work proposed a valid methodology for predicting the speed of a car from dashcam point of view. I follow a neural network architecture paradigm inspired by OpenPilot, formerly themselves inspired by value prediction networks in MuZero. My results demonstrate that this architecture style is viable for a general computer vision task. For future improvements, I would like to input the raw image alongside RAFT's representation for feature extraction. I believe that this would improve model accuracy because colors play an important role in the models awareness, such as detecting lane lines between frames, blacktop, and surrounding treelines. In my current architecture, raw colors are is sacrificed for propagating optical flow data through the network. Lastly, I would like to email my models predictions on the unlabeled video for CommaAI to grade. All relevant code can be found in the project repository [6].

References

[1] Wikipedia contributors. Optical flow — Wikipedia, the free encyclopedia, 2021. URL `https://en.wikipedia.org/w/index.php?title=Optical_flow&oldid=1059399864`. [Online; accessed 15-December-2021].

[2] Zachary Teed and Jia Deng. RAFT: recurrent all-pairs field transforms for optical flow. *CoRR*, abs/2003.12039, 2020. URL `https://arxiv.org/abs/2003.12039`.

[3] Zachary Teed and Jia Deng. princeton-vl/raft. `https://github.com/princeton-vl/RAFT`, 2019.

[4] CommaAI. Commaai/openpilot. `https://github.com/commaai/openpilot`, 2021.

[5] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

[6] Garret Partenza. Commaspeedchallenge. `https://github.com/garrett-partenza-us/CommaSpeedChallenge`, 2021.