Predicting a vehicles velocity using dashcam footage and dense optical flow

Florian Wolf
Department of Mathematics and Statistics, University Konstanz
Konstanz, Germany
florian.2.wolf@uni-konstanz.de

Franz Herbst
Department of Physics, University Konstanz
Konstanz, Germany
franz.herbst@uni-konstanz.de

***Abstract*—In this report bla bla bla**
***Index Terms*—deep learning, computer vision, velocity prediction, dense optical flow**

I. Introduction

Here are some motivational words need

A. Aim of the project

Here we need to clarify the aim of the project

II. Data collection, analysis and preprocessing

For our data set, we used the comma ai speedchallenge[1] data base. This data set provides two dashcam videos: a training video, (20400 frames, shoot at 20 frames per second) including ground truths and a testing video (10798 frames, shoot at 20 frames per second) without labels, which they use for applications to check how well a submitted model is able to generalize. As we only have access to the labels of the test video frames, we decided to split the provided train data by the 80/20 principle into training and testing subsets. Here we did not shuffle the data randomly, as we needed to always have two consecutive frames to calculate the optical flow. We initially used a hard cut off after 80% of the frames, as we wanted to test our model on unseen data, to measure how good our model is able to generalize. We will analyse this naive approach later, when we take a closer look at the results.

A. Data analysis

To analyse the velocity distribution in the two subsets, we plotted the velocity per frame curve in Fig. 1.

The first have of the video mostly represents highway scenarios and the second half only consists of city driving scenes. Therefore we did not expect our model to perform really well using the initial splitting, as the models is mostly trained in highway road traffic scenarios.

B. Preprocessing

Each of the provided frames has a size of $(640, 480, 3)$ pixels. Due to computational limitations, we decided to cut off the last 60 pixels from the lower border, to remove a black frame inside the car, which did not have any effect on the optical flow. Furthermore, we cut the frame size in half and

[1]https://github.com/commaai/speedchallenge

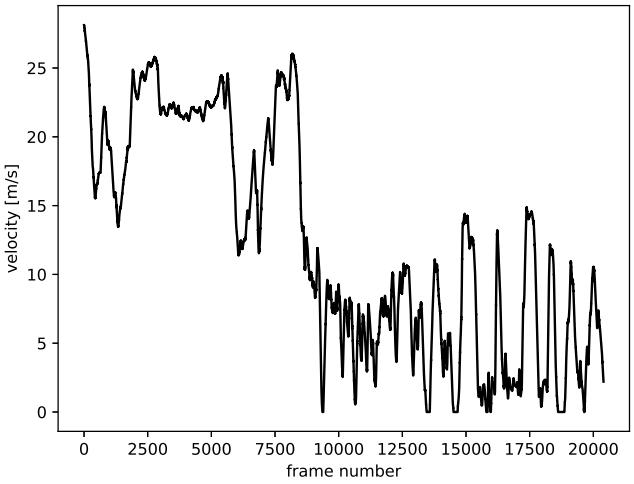


Fig. 1: Distribution of the velocities of all frames in the training video.

calculated the optical flow using the Farneback pyramid [1] method with the following parameters

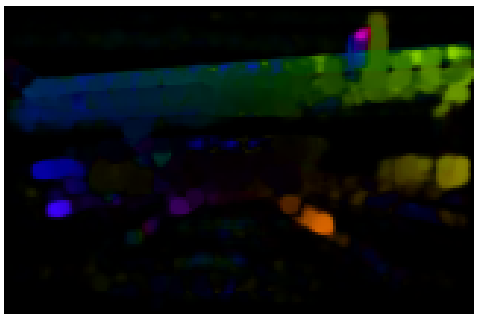
$$\begin{aligned} \text{pyramid levels} &:= 3 \\ \text{pyramid scaling} &:= 0.5 \\ \text{window size} &:= 6 \\ \text{pixel neighborhood size} &:= 5 \\ \text{SD of the gaussian filter} &:= 1.1 \end{aligned}$$

We choose three pyramid level, because we wanted the result to be more accurate. To decrease the training duration, we halved the size of the optical flow frames again, resulting in a resolution of $(160, 105, 3)$ pixels per frame. As we used a window size of 6, a comparison between the original optical flow and the down sampled one lead to the result, that we do not loose a lot of information. As we later on wanted to see if the model performs better using the dashcam frames as additional material, we did the same down sampling with the frames.

Explain, how we display the optical flow.

(a) Original frame

(b) Frame after down sampling



(c) Optical flow field, already down sampled

Fig. 2: Preprocessing of the video frames.

III. Method selection and architecture

Used MSE as performance metric

The prediction of the vehicles speed is a non-linear regression task, the choice of a neural network is reasonable. Recent architectures we discussed in the lecture (RESNET, GOOGLENET, etc.) have shown than using multiple stacked convolution layers combined with stacked dense layers, perform well on image classification tasks. Therefore the choice of a convolutional neural network is justified.

A. Initial Network/prototype

As a base architecture we decided to give the model of https://arxiv.org/pdf/1604.07316v1.pdf a try. As the paper used it for self-driving cars, the model has enough complexity to handle a task like ours and with the amount of layers, we had a lot of possibilities to fine-tune and improve the model.

In a first approach, we tried the raw model gaining an MSE of around 18-20 on testing and under 3 on training.

B. First fine tuning, modifications

Then we used using different activation functions and batch normalization as improvements we discussed in the lecture, to speed up training and improve the performance.

We also used a drop out layer according to https://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf to prevent/reduce overfitting

As proposed in the lecture we used

$$\mathrm{ReLu} : \mathbb{R} \to \mathbb{R}_0^+, x \mapsto \max\{0, x\}$$

as the initial activation function. We used batch normalization according to https://arxiv.org/pdf/1502.03167.pdf, to speed up the training process. Using the ReLU function and XX epochs for training, we achieved a MSE of around 12, running the code multiple times, to ensure this result holds. This result was not really promising, so we wanted to improve the error firstly modifying the model.

Initial splitting	ReLU		leakyReLU	
	Train	Test	Train	Test
No pooling	2.85	12.08	2.45	10.75
Max pooling	5.62	11.82	5.52	10.29
Max pooling (15 epochs)	-	-	**3.22**	**9.63**
Average pooling	7.70	11.40	6.08	13.09

TABLE I: MSE results of the initial network using pooling layers and the leakyReLU activation function.

To solve the problem of dead neurons[2] of the ReLU function, we tried the leakyReLU function

$$\mathrm{leakyReLU} : \mathbb{R} \to \mathbb{R}, x \mapsto \begin{cases} x, x \geq 0 \\ c \cdot x, x < 0 \end{cases}$$

with a parameter $c = 0.01$. Using again batch normalization, we achieved an error of around 10.

We identified three possible reasons for our poor results:

(i) Too complex model, as the paper used it for autonomous driving or we put too little information into the model.

(ii) Problems with different brightnesses in the frames (lack of generality), which leads to unstable calculations in the optical flow, as the optical flow is quite sensitive to brightness. (Explain how in the train part the sky is quite dark and in the city (end) the sky is bright)

(iii) Too naive/ambigous splitting of the data into train and testing set, as both datasets seem to represent totally different scenarios in the road traffic.

We came up with the following approaches to solve these problems

(i) Simplify the model by using Pooling (we will try average and max pooling), to get more compression and we tried on the other hand to feed in more information into the model, by using a linear combination of the optical flow and the raw frames itself, and we tried using a siamese network, to put simultaneously the of and raw frames into the convolutional layers.

(ii) We wanted to try adding some additional noise into the frames before calculating the optical flow, to make the calculation more robust against brightness changes. As intentionally adding noise to a frame is quite atypical in computer vision, this idea looked quite interesting.

(iii) Use another splitting. To get a better ratio between highway and city driving scenarios, we decided to split the data into blocks of 100 frames and take the first 80 for training and the last 20 for testing. Therefore our model should have seen some city driving.

[2]As one can clearly see in the defintion of the ReLU function, neurons with a value below zero cannot participate in the learning process.

ASSUMPTION TRAINING ERROR UNDER 3 IS EXTREMLY GOOD, to avoid overfit

MAYBE PROVIDE TABULAR WITH TRAIN AND TEST ERROR

MISSING ANALYSIS OF THE DATA SET

relatively low number of epochs, as the model always seemed to overfit, because we have a relatively complex model for not a lot of data frames

IV. Results and comparison

V. Further work

Use MUCH MUCH MUCH MUCH MUCH more data!

Create Validation Data that is not connected to training data and still covers all situations

Acknowledgment

We like to thank bla bla bla

References

[1] Gunnar Farnebäck. "Two-Frame Motion Estimation Based on Polynomial Expansion". In: *Scandinavian Conference on Image Analysis* (2003), pp. 363–370.