Predicting a vehicles velocity using dashcam footage

A deep learning approach

Florian Wolf, Department of Mathematics and Statistics
Franz Herbst, Department of Physics

Machine Learning using Matlab
Universität Konstanz

January 19, 2021

Table of content

The "comma ai speed challenge"[1]

Motivation

- HERE ARE SOME MOTIVATIONAL WORDS NEEDED

Data collection:

- "comma ai speed challenge" provides two videos:
  - Train video: 24000 frames, shoot at 20 frames per second, including ground truths
  - Test video: 10798 frames, shoot at 20 frames per second, no ground truths, used to applications
- Split train video after 80% with hard cut off (ability the generalize), to get train and test datasets

Initial assumptions

- Use mean squared error (MSE) as a performance measure
- How to evaluate a prediction? Assumptions:
  - MSE ≤ 10: good
  - MSE ≤ 5: better
  - MSE ≤ 3: correct

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

Preprocessing

- Frame size of $(640, 480, 3)$ pixels
- Cut off last 60 pixels, to remove black frame inside the car
- Sample down the frame to half its size, due to computational limitations

Original frame

Cut off the last 60 pixels, downsampled

Optical flow using “Farneback pyramid method” [2]

- Global method to solve the optical flow equation

$$\partial_x f \cdot V_x + \partial_y f \cdot V_y + \partial_t f = 0$$

  for an image sequence $(f_t)_t$ with $f_t : \Omega \to \mathbb{R}^3$, for all t, and the (dense) flow field $V : \Omega \to \mathbb{R}^2, \omega \mapsto (V_x(\omega), V_y(\omega))$.
- Uses a downsampling pyramid, to solve the equation for different resolutions of the image
- Parameters for the Farneback method

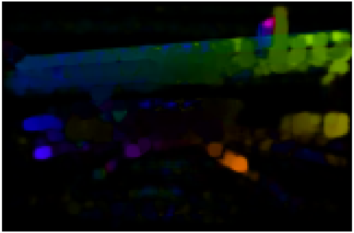
$$\begin{aligned}
\text{pyramid levels} &:= 3 \\
\text{pyramid scaling} &:= 0.5 \\
\text{window size} &:= 6 \\
\text{pixel neighborhood size} &:= 5 \\
\text{SD of the gaussian filter} &:= 1.1
\end{aligned}$$

- Result: **Flow field with** $(160, 105, 3)$ **pixels**

Visualization of the flow field

- Flow field is a two-dimensional vector field
- RGB representation via
  - Transform flow field into polar coordinates $(V_x, V_y) \overset{\simeq}{\mapsto} (r, \varphi)$
  - Normalize magnitudes r for the third channel
  - Values of the second channel are all set to 255
  - Multiply angle φ by factor $\frac{180}{2\pi}$ for the first channel
- Sample down the resolution again, to speed up the training

Original frame



Corresponding flow field, already sampled down

Convolutional neural network and initial architecture

Method selection

- Speed prediction is a **non-linear regression** task $\rightsquigarrow$ Neural network
- Use convolution layers to perform feature extraction $\rightsquigarrow$ **convolutional neural network** (CNN)

Initial architecture

- Paper of NVIDIA work group [1] of a CNN for self-driving cars
- Enough complexity and layers to handle the task and lots of possibilities to fine-tune it
- Initial results with the raw model: MSE of under 3 on the training set and around 18-20 on the testing set
  $\Rightarrow$ Improvements needed

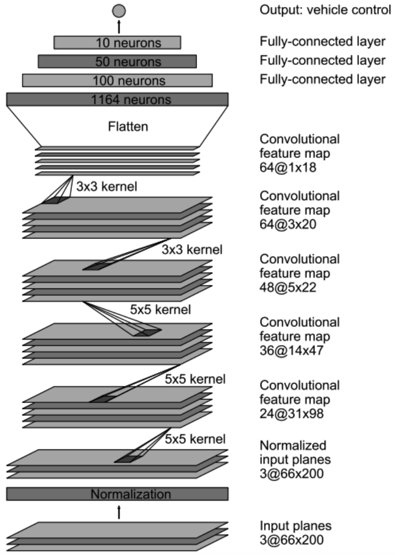


Figure: Original architecture of the NVIDIA paper [1].

Batch Normalization, Dropout layers, activation function and pooling

- Batch normalization to speed up the training [3]
- Initial activation function: $\mathrm{ReLu} : \mathbb{R} \to \mathbb{R}_0^+, x \mapsto \max\{0, x\}$, still MSE of over 15 on the testing set, less then 2 on the training set
  $\Rightarrow$ Overfitting problems
- Found paper about dropout layers [4] to reduce overfit
- Solve problems of dead neurons using

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

  with $c = 0.01$, MSE of around 11 on the testing set and less than 3 on the training set

Problems

We identified three possible problems for our poor results

1. Too complex model, as initially used for autonomous driving or insufficient amount of information put into the model
2. Brightnesses/illumination changes in the frames, therefore unstable calculations of the optical flow
3. Too ambiguous splitting, as the training and testing datasets represent totally different road traffic scenarios

Possible solutions

1. **Simplify model**: pooling layers (maximum and average pooling) to get more compression
   Siamese approach: put flow field and raw frame into the model or put two consecutive frames into the model
2. **Add additional noise**: add noise before computing the optical flow filed, to get more invariance regarding illumination changes
3. **Different splitting**: get better ratio between different scenarios, by using different data splittings: finer one and a more specific one based on the different road traffic situations in the video

New splitting: situational splitting

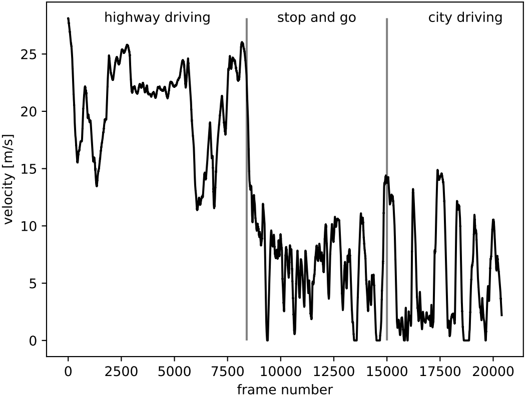


Figure: Velocity distribution in the training video, including labels for different scenarios.

Pooling layers (initial splitting)

Initial splitting, 8 epochs	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: MSE results of the network using different pooling strategies, one dropout layers, two different activation functions and the initial splitting. We trained each of the models for eight epochs.

Siamese approach (new splitting)

RESULTS ARE NEEDED :)

Current work

1. Add additional noise to frames
2. Siamese approach for two consecutive frames
3. Try to figure out why the new splitting does not work

Further ideas: Continuous modelling assumption

THIS IS JUST A TEST

References

Mariusz Bojarski et al. "End to End Learning for Self-Driving Cars". In: (Apr. 2016). URL: https://arxiv.org/pdf/1604.07316v1.pdf.

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

Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: (Feb. 2015). URL: https://arxiv.org/pdf/1502.03167.pdf.

Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.