Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

Национальный исследовательский университет
"Высшая школа экономики"

Факультет компьютерных наук

магистерская работа

Программа для оценки оптического потока на основе подходов глубокого обучения

Студент:
Джозеф Аффрийие Аттакорах

Научный руководитель:
Салех Хади Мухаммед, доцент

Москва, 2020 г.
Government of the Russian Federation

National Research University Higher School of Economics

Faculty of computer science

Master Thesis

The Application for Optical Flow Estimation Based on Deep Learning Approaches

Written by:
Joseph Afriyie Attakorah

Supervisor:
Professor Hadi Saleh

Moscow, 2020

Table of Contents

ABSTRACT

One of the goals of computer science is to enable computers see like humans. This is meant to be achieved through computer vision, a computer science discipline, dedicated to deal with using computers in acquiring, processing, analyzing and understanding digital images [16].

As a goal to determine how computers can be made to gain high-level understanding from digital images and videos and to automate tasks that the human visual system can do, a technique that prioritizes motion as a key characteristic of classification was introduced [7][13][15].

Just as the human visual system has a stimulus for perception of the shape, distance and movement of objects in the real world and control of locomotion, it is also vital for computers to be able to do same or better and have the ability to discern possibilities for action within the environment [17].

Videos come with a great amount of information that can help through this goal but processing them can be highly computational. Given the recent advancement of the deep learning algorithm techniques, this paper seeks to use convolutional neural networks to capture the accurate optical flow from the given input video frames and to further predict the speed of a car from a video input while keeping it computationally feasible within the real-world context such that it can efficiently fit into embedded systems, small and portable devices.

Keywords

Optical flow, sparse technique, dense technique, pixels, deep learning, warp, batch, step size, epoch, activation function, Exponential Linear Unit (ELU), overfitting, early stopping, model, optimizer, convolutional neural network (CNN), displacement

1. INTRODUCTION

Images are great in all; videos are even cooler because we have more information in a video than in an image. Because in an image we have just the spatial positioning of the pixels, thus where they are in the image relative to each other. A video consists of a set of individual film frames or video frames, which are typically many still images, that sequentially compose the complete moving picture to make up the video over time. So, in a video, we get the same spatial information but also have an additional temporal component. Meaning we do not have only the location of a pixel spatially but also when this pixel exists. This additional information gives knowledge about the time and duration of the pixel. In other words, information in a video is encoded not only spatially, but also sequentially and according to a specific order. This amount of information opens a lot of doors on what we can look into and makes classifying videos more interesting. Optical flow is one of these doors. Optical flow is computer vision technique use to track the apparent motion of objects in a video. This technique has a number of different applications including video compression, video stabilization, video description (a more recent application area), object detection and tracking (e.g. detect and track vehicles in an automated traffic surveillance application), velocity estimation, and to mention but a few.

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[5]. Optical flow can also be defined as the distribution of apparent velocities of movement of brightness pattern in an image [12]. Optical flow is a per-pixel prediction which means to estimate how the pixel brightness moves across the screen over time [10].

With the recent advancements in AI, Deep nets are becoming a popular approach to solving problems. In this project, we try to achieve an algorithm that prioritizes motion as a key characteristic of classification and is computationally feasible within the real-world context. The best possible way is through optical flow

Deep learning is part of a broader family of machine learning methods based on artificial neural networks that uses multiple layers to progressively extract higher level features from the raw input. For example, in image processing, lower layers may identify edges, while higher layers may identify the concepts relevant to a human such as digits or letters or faces. Learning can be supervised, semi-supervised or unsupervised.

Determining the speed of a car can be very challenging but with labelled data and a CNN architecture and with help of optical flow, we might be able to estimate the speed.

1.1. BACKGROUND

The concept of optical flow was introduced by the American psychologist James J. Gibson in the 1940s to describe the visual stimulus provided to animals moving through the world. Gibson stressed the importance of optic flow for affordance perception, the ability to discern possibilities for action within the environment [11].

We live in a moving world. Perceiving, understanding and predicting motion is an important part of our daily lives.

As a goal to determine how computers can be made to gain high-level understanding from digital images and videos and to automate tasks that the human visual system can do, optical flow became a computer vision technique with pioneering researchers Horn and Schunck.

Now we cannot talk about optical flow without mentioning motion. Motion estimation is a relevant task of video analysis. It can be used to find the motion fields, to identify moving objects, calculate object path or trajectory and to find their speed. However, many of the methods developed for the motion analysis are computationally expensive or require already segmented picture.

In this paper, we present an efficient method for computing direction of motion of a vehicle and estimate its speed. The proposed method firstly uses optical flow algorithm to calculate changes in the intensity of the pixels of the images. These apparent velocity components are then subjected to image processing techniques to obtain centroid of the vehicle across the frames. The image coordinates of the centroid are mapped to World space. Using this information, the velocity of the vehicle is estimated. In the following sections, we briefly survey the relevant literature, describe our method and study the experiments performed

Optical-flow methods are based on computing estimates of the motion of the image intensities over time in a video. The flow fields can then be analyzed to produce segmentations into regions, which might be associated with moving objects [14]. Optical-flow or motion-estimation algorithms can be used to detect and delineate independently moving objects, even in the presence of camera motion. Of course, optical-flow-based techniques are computationally complex, and hence require fast hardware and software solutions to implement. Since optical flow is fundamentally a differential quantity, estimation of it is highly susceptible to noise; ameliorating the noise sensitivity can imply increases in complexity [14]

There are quite a few deep learning algorithms that are applied to the spatial domain ranging from classification, segmentation, scene understanding etc. However, algorithms that perform well in the temporal domain are fewer and less developed than their spatial counterpart. This is because of the complicated nature of adding time to the equation. Optical flow is a desired computer vision technique that prioritizes motion as a key characteristic of classification and is computationally feasible in the real-world context.

1.2. Optical Flow Estimation

In a typical scene like one below we might want to estimate the motion of the vehicles and track their direction as well as their speed.

Figure 1. Motion Scene [6].

Sequences of ordered images allow the estimation of motion as either instantaneous image velocities or discrete image displacements. The optical flow methods try to calculate the motion between two image frames which are taken at times **t** and **t + Δt** at every voxel position

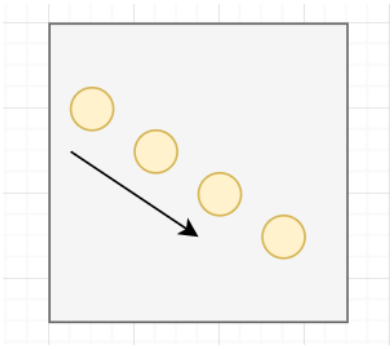

Figure 2.a. Direction of Object.

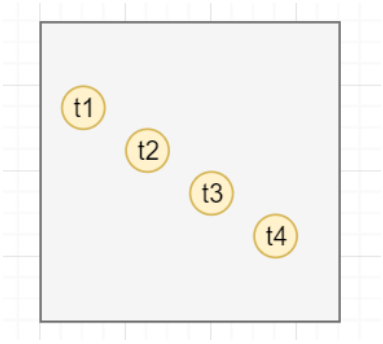

Figure 2.b. Time sequence of Object.

To make an Intuition behind optical flow, let's consider the above diagram to depict an object, say a ball, thrown from the upper left corner of the scene to the lower right corner of the scene. The t1…t4 represent the balls position at specific times, sequentially.

If we want to track an object through a video, in a computer level, we only have access to the raw pixels. We take individual frames look at the pixels and track them in successive frames as depicted below in the diagram.

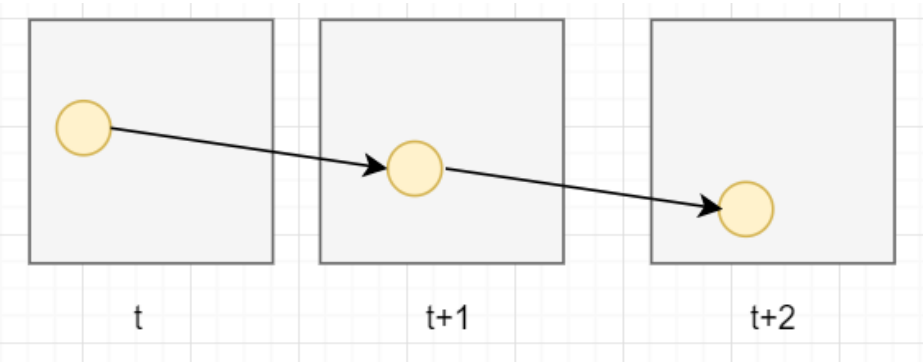

Figure 3. Displacement across frames.

We maintain the assumption that the observed brightness of any object point is constant over time. Implying that nearby points in the image plane move in a similar manner or move together, and pixel intensities don't rapidly change between consecutive frames thus pixel values in two successive frames don't immediately change.

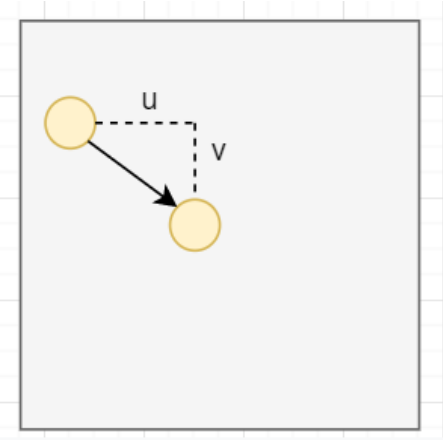

Figure 4. Determining Displacement.

So, the problem of flow here is that the pixel moved in the x direction by some amount **u** and down in the y direction by some amount **v**. So, the challenge of flow is to find the **u** and **v** that it moved, which is equal to the displacement depicted by the arrow. With that we can track the path. This is what optical flow seeks to solve.

The mathematics involved has been handled by pioneering researchers, but their solutions are computationally expensive.

From that time, the problem of optical flow has historically been an optimization problem in computer vision. Since optical flow is used in motion detection, video compression and several safety-critical applications like self-driving cars, it is important to optimize it to the best of its performance with state-of-the-art.

The application of deep learning techniques in recent times has shown impressive results. Compared with traditional methods, CNN achieved a large improvement in quality in estimating optical flow.

1.3. Vehicle Speed Estimation

Since a great amount of information can be gathered from a video, and optical flow can help in estimating the apparent motion of an object in a video sequence, we might be able to determine the speed of a moving car from a dashboard video. Being able to achieve this can enable us to expand to other use cases like in sports, such as determining the speed of a athletes in race in real time from a body camera video.

1.4. Research Question

How well do neural networks perform at predicting the speed of a car given a dashcam video?

1.5. Objective

To increase the accuracy of estimating the speed of a moving car from video.

We plan to be achieved this by using optical flow and deep learning approaches and by minimizing the validation mean square error of the model while increasing the model accuracy.

2. STATE OF THE ART (LITERATURE REVIEW)

After the American psychologist James J. Gibson in the 1940s introduced the concept of optical flow, followers and other researchers further developed on this concept, eventually establishing this concept as a computer vision technique. The following are the methods proposed by some of these early researchers for finding optical flow:

2.1. Methods for Determination

- Differential methods of estimating optical flow, based on partial derivatives of the image signal and/or the sought flow field and higher-order partial derivatives, such as:
  - Lucas–Kanade method – regarding image patches and an affine model for the flow field
  - Horn–Schunck method – optimizing a functional based on residuals from the brightness constancy constraint, and a particular regularization term expressing the expected smoothness of the flow field
  - Buxton–Buxton method – based on a model of the motion of edges in image sequences
  - Black–Jepson method – coarse optical flow via correlation [3]
  - General variational methods – a range of modifications/extensions of Horn–Schunck, using other data terms and other smoothness terms.
- Discrete optimization methods – the search space is quantized, and then image matching is addressed through label assignment at every pixel, such that the corresponding deformation minimizes the distance between the source and the target image [4].

2.2. Summary of Methods and Algorithms

The following table summarizes the previous and current state of the art:

Technology	Year	Advantages	Limitations
Determining optical flow. Artificial intelligence, Horn, B. K., & Schunck	1981	yields a high density of flow vectors, i.e. the flow information missing in inner parts of homogeneous objects is filled in from the motion boundaries	complex and computationally expensive in real-time applications. It is more sensitive to noise than local methods
Lucas Kanade method regarding image patches and an affine model for the flow field	1981	very fast calculation and accurate time derivatives	Manually tune all parameters, hence low scalability and errors on boundaries of moving object
FlowNetCorr	2015	Faster rendering speed than traditional computer vision methods	Two images are convoluted separately and are combined by a correlation layer. Poor performance on displacement
Flownet: Learning optical flow with convolutional networks. (**FlowNetS**)	2015	Faster rendering speed than traditional computer vision methods	simply stacks two sequentially adjacent images as input. Produces noisy and blurred results
FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks	2016	stacked by flownets and flownetcorr, and produces much better results. Based on FlowNet	FlowNet 2.0 is only marginally slower than the original FlowNet although decreases the estimation error by more than 50%
Optical Flow Estimation using a Spatial Pyramid Network (**Spynet**)	2016	much more lightweight, with fewer parameters, faster, and produces large motions in a course to fine manner	performs below FlowNet2.0

Table 1. Summary of Methods and Algorithms.

Ratings of the above methods for suitability for mobile devices:

Algorithm	**Model Size**	**Accuracy**	**Mobile Support (Average)**
SpyNet	4	3	3.5
FlowNet2.0	2	4	2
FlowNetCorr	3	2	2.5
FlowNetS	3	2	2.5

Table 2. Ratings.

Heavyweight 1 - Lightweight 5

Low accuracy 1 – High accuracy 5

Bad Mobile support 1 – Good Mobile support 5

2.3. Further Details on Related works

Optical flow is the pattern of apparent motion of image objects between two consecutive frames caused by the movement of object or camera. It is 2D vector field where each vector is a displacement vector showing the movement of points from first frame to second.

2.3.1. Lucas kanade

The Lucas-Kanade optical flow algorithm is the most widely known traditional technique which can estimates the movement of interesting features in successive images of a scene. This algorithm is base on some assumptions such as that the two images are separated by a small-time increment Δt, in such a way that objects have not displaced significantly (that is, the algorithm works best with slow moving objects). [2]. The Lucas-Kanade algorithm makes a “best guess” of the displacement of the surroundings by considering changes in pixel intensity which can be explained from the known intensity gradients of the image in that surroundings. This is done using the least squares criterion. The outcome of the algorithm is a set of optical flow vectors which are distributed over the image to provide an estimation idea of the movement of objects in the scene. This works for works for moderate object speeds. Assumption of constant flow (pure translation) for all pixels in a larger window is unreasonable for long periods of time

2.3.2. Horn and Schunck

Horn and Schunck presented a method for finding the optical flow pattern which assumes that the apparent velocityty of the brightness pattern varies smoothly almost everywhere in the image. It tries to minimize distortions and takes solutions which shows smoothness. While this method is more sensitive to noise, the flow information missing in the inner parts of homogeneous object is filled in from the motion boundaries [12].

2.3.3. FlowNet

Given the recent advancement in machine learning, precisely convolutional neural network (CNN) which has tremendously been successful in a variety of computer vision tasks, FlowNet sought to build on recent progress in design of convolutional neural network architecture and improve on optical flow estimation. Convolutional neural networks are known to be very good at learning input–output relations given enough labelled data [FlowNet]. FlowNet proposes end to end optical flow estimation using CNN that takes in two input images and outputs the flow field.

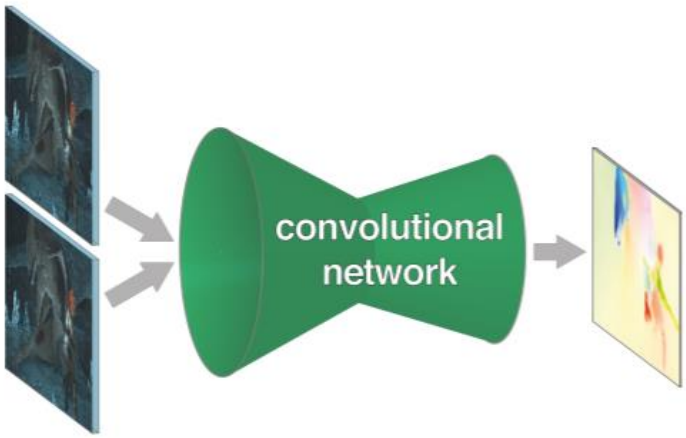

Figure 5. Neural Network model for optical flow [8].

2.3.4. FlowNet2.0

FlowNet2.0 advances the concept of end-to-end learning of optical flow and makes it work well. It used three major contributions to caused large improvements in quality and speed; focusing on the training data with much attention to the schedule of presenting data during the training, developing a stacked architecture that includes warping of the second image with intermediate optical flow and elaborating on small displacements by introducing a subnetwork specializing on small motions [9]. This idea made it achieve better accuracy than its predecessor, FlowNet.

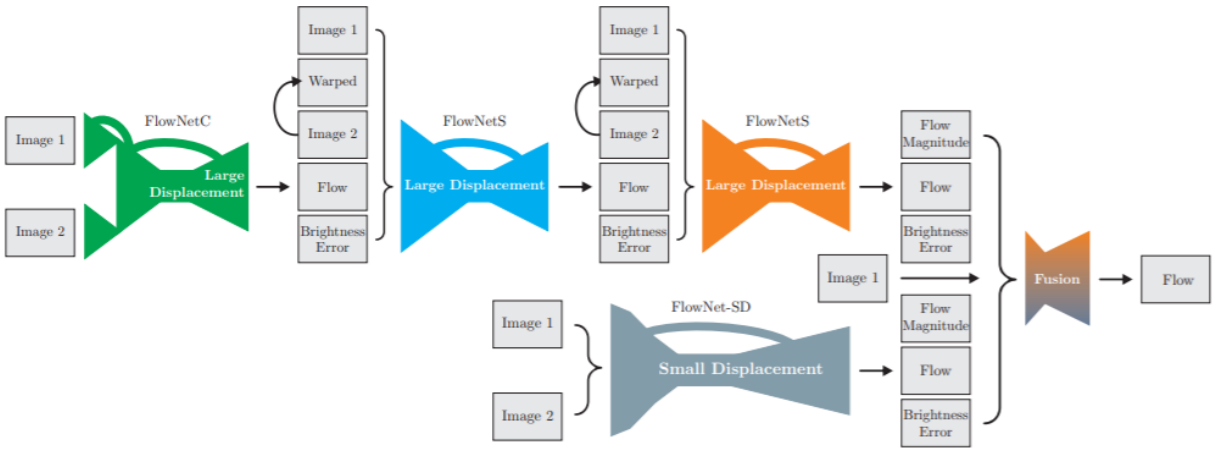

Figure 6. FlowNet Architecture [9].

2.3.5. SPYNET

Decades of research on optical flow has produce well engineered systems but there has always been more room for improvements. Some try to enhance on the performance of others while others seek to improve on the accuracy. SpyNet does not only seek to achieve the combination of the above but to also reduce memory requirement to make flow more practical for embedded

systems, robots and mobile applications [1]. SpyNet uses the coarse-to-fine spatial pyramid approach to learn residual flow at each pyramid level.

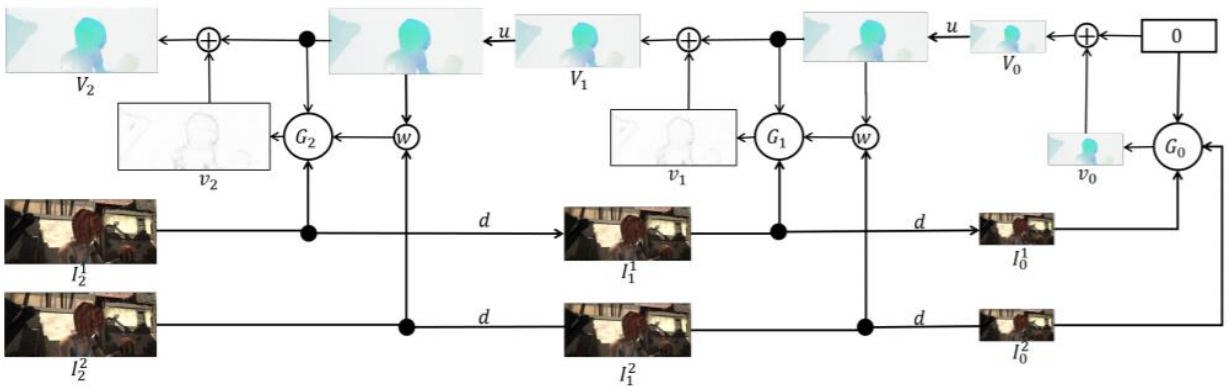

Figure 7. SpyNet architecture [1].

The evaluation of SpyNet on all the standard benchmarks found that SpyNet is the most accurate overall, with and without fine tuning [1]

Method	Sintel Clean		Sintel Final		KITTI		Middlebury		Flying Chairs	Time (s)
	train	test	train	test	train	test	train	test	test	
Classic+NLP	4.13	6.73	5.90	8.29	-	-	0.22	0.32	3.93	102
FlowNetS	4.50	7.42	**5.45**	**8.43**	**8.26**	-	1.09	-	2.71	0.080
FlowNetC	4.31	7.28	5.87	8.81	9.35	-	1.15	-	**2.19**	0.150
SPyNet	**4.12**	**6.69**	5.57	**8.43**	9.12	-	**0.33**	**0.58**	2.63	**0.069**
FlowNetS+ft	3.66	6.96	4.44	**7.76**	7.52	9.1	0.98	-	3.04	0.080
FlowNetC+ft	3.78	6.85	5.28	8.51	8.79	-	0.93		**2.27**	0.150
SPyNet+ft	**3.17**	**6.64**	**4.32**	8.36	8.25	10.1	**0.33**	**0.58**	3.07	**0.069**
SPyNet+ft*	-	-	-	-	**3.36**	**4.1**	-	-	-	-

Table 3. Average end point errors (EPE). Results are divided into methods trained with (+ft) and without fine tuning. SPyNet+ft* uses additional training data compared to FlowNet+ft. Bold font indicates the most accurate results among the convnet methods. All run times are measured on Flying Chairs and exclude image loading time [1].

Method	Sintel Final						Sintel Clean					
	$d_{0\text{-}10}$	$d_{10\text{-}60}$	$d_{60\text{-}140}$	$s_{0\text{-}10}$	$s_{10\text{-}40}$	s_{40+}	$d_{0\text{-}10}$	$d_{10\text{-}60}$	$d_{60\text{-}140}$	$s_{0\text{-}10}$	$s_{10\text{-}40}$	s_{40+}
FlowNetS+ft	7.25	4.61	**2.99**	1.87	5.83	43.24	5.99	3.56	2.19	1.42	3.81	40.10
FlowNetC+ft	7.19	4.62	3.30	2.30	6.17	**40.78**	5.57	3.18	1.99	1.62	3.97	**33.37**
SpyNet+ft	**6.69**	**4.37**	3.29	**1.39**	**5.53**	49.71	**5.50**	**3.12**	**1.71**	**0.83**	**3.34**	43.44

Table 4. Comparison of FlowNet and SpyNet on the Sintel benchmark for different velocities, s, and distances, d, from motion boundaries [1].

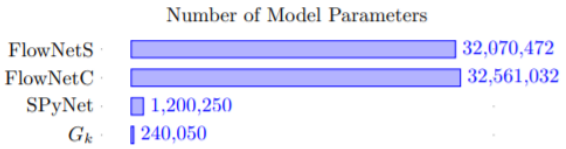

Figure 8. Model size of various methods. SpyNet model is 96% smaller than FlowNet [1].

2.3.6. Vehicle tracking and speed estimation using optical flow

Indu, Gupta and Bhattacharyya presented a presented a paper on estimating the speed of a car from a fixed mounted camera where the vehicle motion is detected and tracked using traditional Horn-Schunck method. In their approach, they first calculate the distance travelled by the vehicle by using the motion of the centroid over the successive frames and then the speed is estimated. Coordinates of the centroid in successive frames are mapped to real world coordinates to calculate the Euclidean distance. Then they obtain the time of motion in seconds. With the distance determined, and the time of motion obtained, they calculate the velocity of the car [18].

While they achieved a fairly low percentage error of 0.98% the shortfall of their approach is that it can only be applied to a fixed setting which limits its range of use-cases.

3. TECHNOLOGIES

1. Python
2. TensorFlow
3. Keras
4. Git
5. GitHub
6. Pytorch

4. APPROACH

The general approach might be to train a network with the video frames and the labelled data. But since there are no common features to associate with speed, the model will very likely overfit by memorizing the images and the labelled speed values.

Using Optical flow estimation, we can find the pixel displacements and generate some other features which can better help us through with the objective of estimating the speed of a car from a video.

In this approach we will create a convolutional neural network model that takes optical flow output as input and estimate the speed of a car. The optical flow estimation model is pre-trained with Sintel clean dataset. Our input data which is a video is first broken into frames and pre-processed, then converted to optical flow. Afterwards, passed to the Convolutional Neural network and trained against the labels available in the dataset. The diagrams below depict our approach:

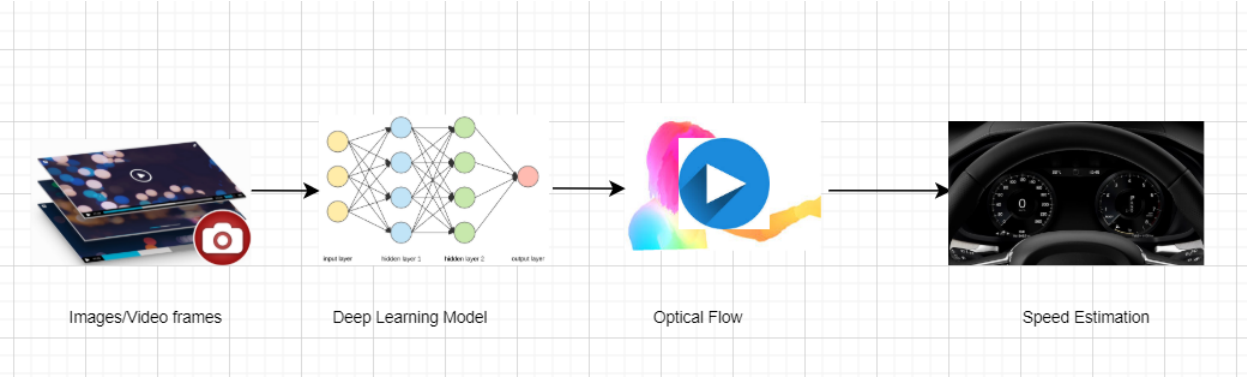

Figure 9.a. Solution approach.

Further;

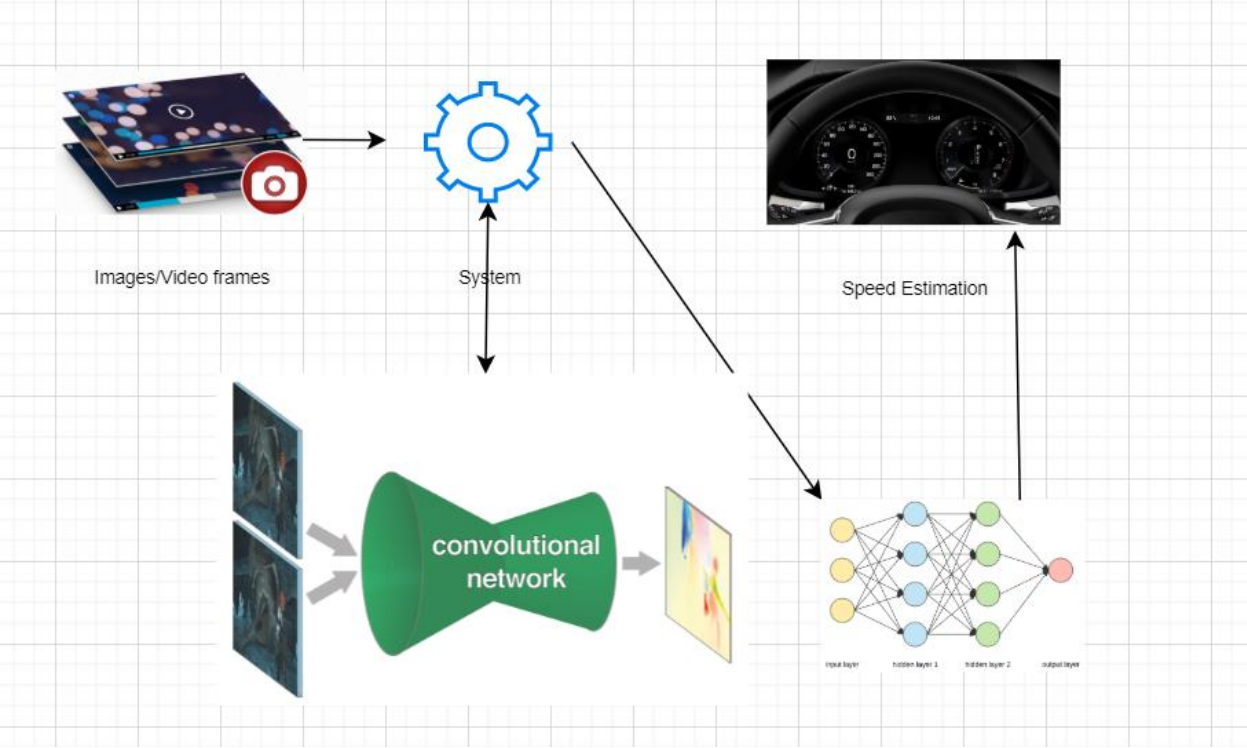

Figure 10.b. Solution approach

4.1. Dataset

Images from the Sintel dataset is used for optical flow.

The dataset for predicting the speed of a car from a video is acquired from the comma.ai Programming Challenge which it includes

- data/train.mp4 is a video of driving containing over 20000 frames. Video is shot at 20 fps.
- data/train.txt contains the speed of the car at each frame, one speed on each line.
- data/test.mp4 is a different driving video containing 10798 frames. Video is shot at 20 fps.

Breaking them down:

Total Images: 20696

Train samples: 16556

Validation samples: 4140

Number of labels: 20696

4.2. Solution Approach in Steps

1. Break down the video into individual frames (480x640)
2. Adjust the image to minimize the amount of unnecessary dynamic and static background that may impact the discernment of relevant features.
    i. Apply a brightness contrast
    ii. Determine the Region of Interest (ROI)
3. Convert to optical flow images
4. Convert the flo files to colour
5. Resize the images (240x320)
6. Divide the dataset into training and validation dataset (15%)
7. Feed it to the CNN for training
8. Predict the speed of car in the test video

Our model is represented with an input layer, several hidden layers each with Exponential Linear Unit (ELU) activation function to determine when a neuron should be activated and an output layer

We start with a learning rate and how aggressively our model would try to find the best model of 0.0001

During training, the loss will be monitored and when the discrepancy between the desired output and the actual output start to rise, we stop training by setting early stopping with patience of 3 (three)

We use the Mean Squared Error loss function and Adam optimizer, a best practice optimizer to change the internal variables a bit at a time to gradually reduce the loss function

Some starting configurations include:

Batch size: 96

Epoch: 200

Training is done on ubuntu 18.04 with a NVIDIA GEFORCE GTX 1060 Cuda Enabled GPU using the NVIDIA CNN architecture for End to End learning for self-driving cars which is continually adjusted for better results.

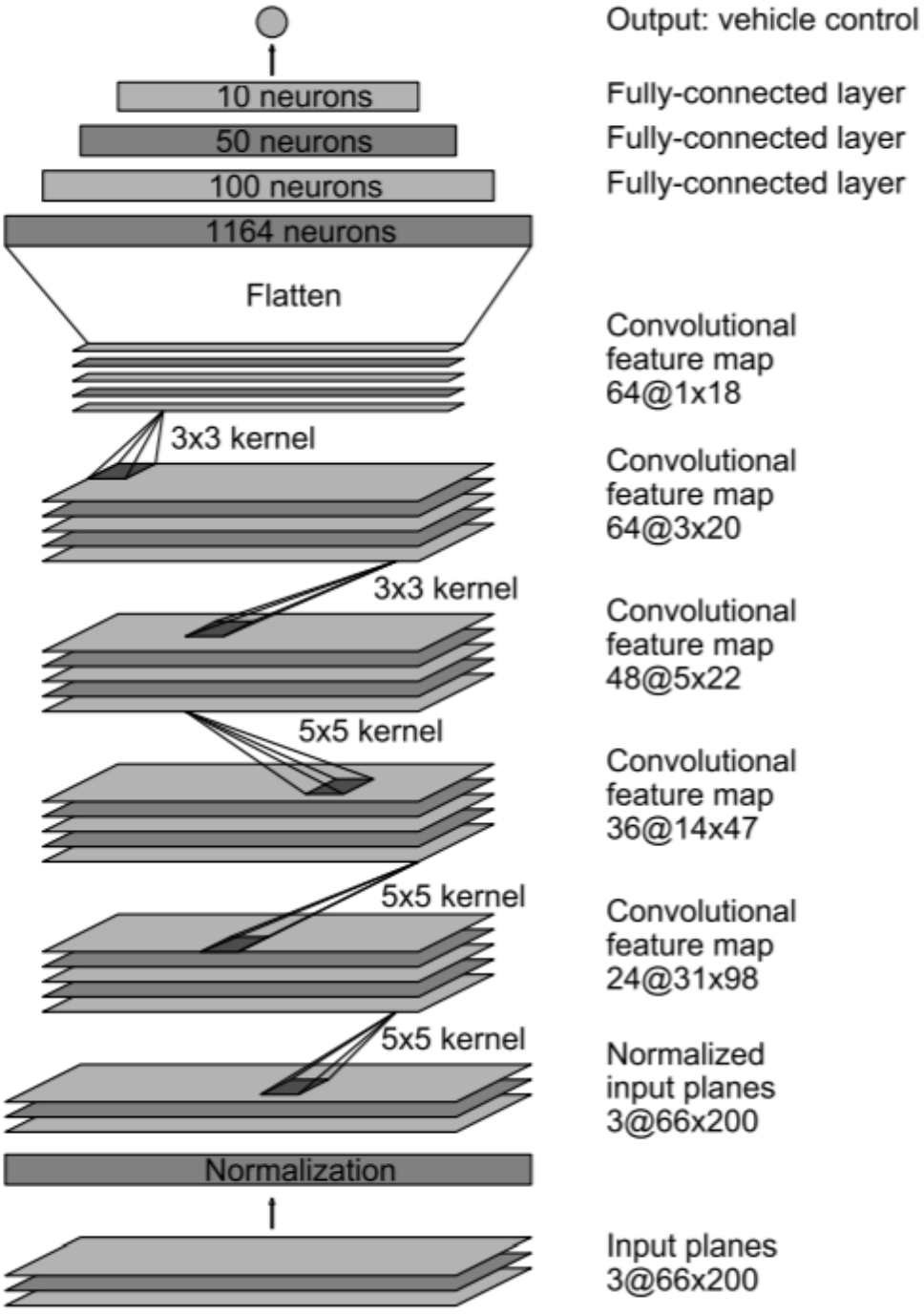

Figure 11. NVIDIA CNN architecture for End to End learning for self-driving cars [19]

5. RESULTS

Current result

```
129/129 [==============================] - 811s 6s/step - loss: 1.9403 - accuracy: 0.0129 - val_loss: 2.2383 - val_accuracy: 0.0128
Epoch 6/50
129/129 [==============================] - 812s 6s/step - loss: 1.4784 - accuracy: 0.0162 - val_loss: 3.0822 - val_accuracy: 0.0168
Epoch 7/50
129/129 [==============================] - 816s 6s/step - loss: 1.1998 - accuracy: 0.0179 - val_loss: 2.1463 - val_accuracy: 0.0111
Epoch 8/50
129/129 [==============================] - 850s 7s/step - loss: 0.8843 - accuracy: 0.0196 - val_loss: 1.3973 - val_accuracy: 0.0171
Epoch 9/50
129/129 [==============================] - 813s 6s/step - loss: 0.7460 - accuracy: 0.0236 - val_loss: 1.4340 - val_accuracy: 0.0146
Epoch 10/50
129/129 [==============================] - 812s 6s/step - loss: 0.5783 - accuracy: 0.0284 - val_loss: 1.5054 - val_accuracy: 0.0252
Epoch 11/50
129/129 [==============================] - 814s 6s/step - loss: 0.4954 - accuracy: 0.0294 - val_loss: 1.3914 - val_accuracy: 0.0207
Epoch 12/50
129/129 [==============================] - 812s 6s/step - loss: 0.3844 - accuracy: 0.0321 - val_loss: 0.8892 - val_accuracy: 0.0254
Epoch 13/50
129/129 [==============================] - 814s 6s/step - loss: 0.3549 - accuracy: 0.0334 - val_loss: 1.1118 - val_accuracy: 0.0277
Epoch 14/50
129/129 [==============================] - 866s 7s/step - loss: 0.3083 - accuracy: 0.0369 - val_loss: 1.3754 - val_accuracy: 0.0262
Epoch 15/50
129/129 [==============================] - 828s 6s/step - loss: 0.2916 - accuracy: 0.0366 - val_loss: 0.9199 - val_accuracy: 0.0309
Training model complete...
Loss:
[22.637975589249486, 6.057266863471607, 3.826563205746859, 2.5958866102433653, 1.9379262302459985, 1.4783853140369598, 1.1969046183157235, 0.884397748484935
Validation Loss:
[9.26354694366455, 5.645624160766602, 2.899700403213501, 3.4370741844177246, 2.238281011581421, 3.0822341442108154, 2.1463029384613037, 1.3973114490509033,
```

Figure 12. Training result 1

After some adjustments,

```
Epoch 9/50
129/129 [==============================] - 807s 6s/step - loss: 0.4536 - accuracy: 0.0355 - val_loss: 2.2684 - val_accuracy: 0.0289
Epoch 10/50
129/129 [==============================] - 812s 6s/step - loss: 0.3388 - accuracy: 0.0374 - val_loss: 1.1933 - val_accuracy: 0.0278
Epoch 11/50
129/129 [==============================] - 806s 6s/step - loss: 0.3061 - accuracy: 0.0365 - val_loss: 1.2252 - val_accuracy: 0.0349
Epoch 12/50
129/129 [==============================] - 819s 6s/step - loss: 0.2452 - accuracy: 0.0381 - val_loss: 1.5425 - val_accuracy: 0.0309
Epoch 13/50
129/129 [==============================] - 817s 6s/step - loss: 0.2051 - accuracy: 0.0379 - val_loss: 0.9355 - val_accuracy: 0.0294
Epoch 14/50
129/129 [==============================] - 816s 6s/step - loss: 0.1813 - accuracy: 0.0373 - val_loss: 0.8093 - val_accuracy: 0.0356
Epoch 15/50
129/129 [==============================] - 805s 6s/step - loss: 0.1448 - accuracy: 0.0399 - val_loss: 0.9722 - val_accuracy: 0.0233
Epoch 16/50
129/129 [==============================] - 805s 6s/step - loss: 0.1256 - accuracy: 0.0414 - val_loss: 1.2023 - val_accuracy: 0.0381
Epoch 17/50
129/129 [==============================] - 806s 6s/step - loss: 0.1237 - accuracy: 0.0391 - val_loss: 1.1129 - val_accuracy: 0.0310
Epoch 18/50
129/129 [==============================] - 811s 6s/step - loss: 0.1184 - accuracy: 0.0397 - val_loss: 0.8533 - val_accuracy: 0.0350
Epoch 19/50
129/129 [==============================] - 807s 6s/step - loss: 0.1022 - accuracy: 0.0405 - val_loss: 0.6190 - val_accuracy: 0.0356
Training model complete...
Loss:
[22.612269493960596, 5.1265503061581095, 3.1223602885888795, 2.062026229948495, 1.4153614181772125, 0.9844087429048953, 0.7411841639846614, 0.5344667656885
Validation Loss:
[5.89899206161499, 4.254426002502441, 2.9206221103668213, 1.8182547092437744, 1.905908465385437, 1.5858052968978882, 1.343275785446167, 1.0416144132614136,
```

Figure 13. Training result 2

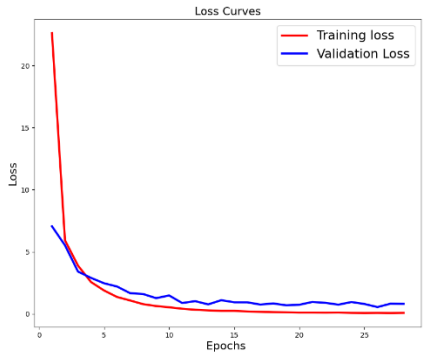

Figure 13. Loss Curves

Figure 14. Prediction

9. BIBLIOGRAPHY (REFERENCES)

[1] Anurag Ranjan and Michael J. Black, 2016, Optical Flow Estimation using a Spatial Pyramid Network, arXiv:1611.00850v2

[2] B.D. Lucas, T. Kanade, ”An Image Registration Technique with an Application to Stereo Vision”, in Proceedings of Image Understanding Workshop, 1981, pp. 121-130

[3] Beauchemin, S. S.; Barron, J. L. (1995). The computation of optical flow. ACM New York, USA.

[4] B. Glocker; N. Komodakis; G. Tziritas; N. Navab; N. Paragios (2008). Dense Image Registration through MRFs and Efficient Linear Programming

[5] Burton, Andrew; Radford, John (1978). Thinking in Perspective: Critical Essays in the Study of Thought Processes. Routledge. ISBN 978-0-416-85840-2.

[6] Chuan-en Lin, 2019, Introduction to Motion Estimation with Optical Flow, Retrieve from https://nanonets.com/blog/optical-flow/

[7] Dana H. Ballard; Christopher M. Brown (1982). Computer Vision. Prentice Hall. ISBN 978-0-13-165316-0.

[8] Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., ... & Brox, T. (2015). Flownet: Learning optical flow with convolutional networks. In Proceedings of the IEEE international conference on computer vision (pp. 2758-2766).

[9] Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., & Brox, T. (2017). Flownet 2.0: Evolution of optical flow estimation with deep networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2462-2470).

[10] Gituma, Mark. “What Is Optical Flow and Why Does It Matter in Deep Learning.” Medium, The Startup, 23 June 2019, medium.com/swlh/what-is-optical-flow-and-why-does-it-matter-in-deep-learning-b3278bb205b5.

[11] G. Johansson, “Visual Perception of Biological Motion and a Model For Its Analysis", Perception and Psychophysics 14, 201-211, 1973

[12] Horn, Berthold K.P.; Schunck, Brian G. (August 1981). "Determining optical flow" (PDF). Artificial Intelligence. 17 (1–3): 185–203. doi:10.1016/0004-3702(81)90024-2. hdl:1721.1/6337

[13] Huang, T. (1996-11-19). Vandoni, Carlo, E (ed.). Computer Vision : Evolution And Promise (PDF). 19th CERN School of Computing. Geneva: CERN. pp. 21–25. doi:10.5170/CERN-1996-008.21. ISBN 978-9290830955.

[14] Joonsoo Lee Al Bovik, 2009, The Essential Guide to Video Processing (Second Edition), CHAPTER 19 - Video Surveillance, https://doi.org/10.1016/B978-0-12-374456-2.00022-0

[15] Milan Sonka; Vaclav Hlavac; Roger Boyle (2008). Image Processing, Analysis, and Machine Vision. Thomson. ISBN 978-0-495-08252-1.

[16] Reinhard Klette (2014). Concise Computer Vision. Springer. ISBN 978-1-4471-6320-6.

[17] Royden, C. S.; Moore, K. D. (2012). "Use of speed cues in the detection of moving objects by moving observers". Vision Research. 59: 17–24. doi:10.1016/j.visres.2012.02.006

[18] S.Indu, Manjari Gupta and Prof. Asok Bhattacharyya, 2011. Vehicle tracking and speed estimation using optical flow method. ISSN: 0975-5462

[19] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, Karol Zieba, (2016), End to End Learning for Self-Driving Cars, arXiv:1604.07316