USING EYE GAZE TRACKING AND IMITATION LEARNING TO IMPROVE AUTONOMOUS DRIVING

A THESIS
SUBMITTED TO
THE DEPARTMENT OF ELECTRONICS AND ELECTRICAL COMMUNICATIONS
FACULTY OF ENGINEERING, CAIRO UNVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
BACHELOR OF ENGINEERING

ALAA OSAMA ABD ELFATTAH
DALIA ANWAR SAYED
MUHAMMAD KHALED ABBAS
MOHAMED ABD ELKARIM SHABAN
MOHAMED ABD ELALIM
MOSTAFA MOHAMED HASSAN

UNDER SUPERVISION OF
DR. HASSAN MOSTAFA DR. SAMAH ELSHAFIEY

JULY 2020

Abstract

Imitation learning is the concept of mimicking the behavior of the expert human driver which is successfully implemented using end-to-end systems. This technique has been widely used because it optimizes all processing steps simultaneously and achieves very good results. A beginner human driver learning from an expert does not only observes his steering actions but also, he is observing his full-body actions and behaviors. We argue that giving an end-to-end model more information about the human driver state during driving such as human gaze behavior, which contains a huge amount of information and is the most important sense in the task of driving, can improve the model performance significantly.

We investigate multiple experiments and introduce a novel architecture that incorporates front-facing camera frames and gaze information into an end-to-end model which achieves a state of the art performance in the task of lane following.

Our architecture uses a spatial transformer network and a multitask network to make steering angle predictions as well as predicting the gaze maps for an input frame in real-time. The model can generalize to different driving environments without being explicitly trained in them.
We also perform a road-test on a recorded dataset from our streets proving that the model can perform in different environments.

Acknowledgment

This dissertation would not have been possible without the support of many people. First and foremost, we would like to thank our advisors Dr. Hassan Mostafa, Dr. Samah El-Shafiey, Dr. Ibrahim Sobh, and Dr. Mohamed Abdou.

We would like to thank Dr. Hassan Mostafa and Dr. Samah El-Shafiey for their encouragement, guidance, the great supervision they offered us throughout the whole project, and the equipment they helped us get access to.

We would like to thank Dr. Ibrahim Sobh for his perfect technical guidance throughout the whole project, which helped us reach our desired objective, address various complex problems, and implement several advanced techniques. We would specially thank him for his patience and the tremendous amount of support and guidance he provided us...

Contents

List of Tables

List of Figures

Chapter 1
Introduction

Countries around the world have responded to the COVID-19 coronavirus with lockdowns, restrictions, and technology solutions that use artificial intelligence to combat the virus. As the world begins to emerge from the pandemic, China is first to emerge from COVID-19 imposed lockdowns thanks to cutting-edge technology, with autonomous vehicles and smart cities seeing an acceleration during this time.

In China, new opportunities for the autonomous driving industry and intelligent solutions have stood out. Restrictions on retail, dining, and everyday life during the outbreak have increased demand for driverless deliveries and non-contact operations, both heavily relying on autonomous driving technologies.

Autonomous driving has also proved to be essential in the fight against the pandemic, easing the burden of COVID-19 by transporting necessary medical supplies and food to health-care professionals and the public in infected areas and disinfecting hospitals and public surfaces to reduce the spread of coronavirus. Simultaneously, the vehicle can serve as a night-time security robot and create alerts about those who are disregarding the coronavirus prevention guidelines, such as not wearing masks or gathering in large crowds.

There are various benefits self-driving cars can offer on different aspects. Most importantly, they could make roads much safer. The leading cause of most accidents in our daily life is human error. According to the statistics provided by the World Health Organization (WHO) [1], most of the road fatalities are caused by human error, therefore self-driving cars can provide a reliable approach for reducing these human errors.

Moreover, there is a study made by Eno Centre for Transportation [2], this study found out that if ten percent of all cars were self-driving, as many as 211,000 accidents would be prevented annually. Some 1,100 lives would be preserved, and the economic costs of automobile accidents would be reduced by more than $20 billion. An additional benefit could be decreasing or even eliminating traffic congestion which can be achieved by self-driving cars by following a consistent behavior during traffic jams, turning all cars on the road into a fleet of cars moving similarly with interconnection and intercommunication among them.

Another crucial aspect is the amount of time and effort spent during driving daily, but with self-driving cars drivers can take over the whole driving task, letting drivers make use of their time. Also, self-driving cars could come in handy in emergencies. For example, if a driver lost consciousness, a vehicle equipped with self-driving technology could take them to safety.

Taxonomy of Driving Automation

It describes the level of automation in a driving system, there are some things we need to take into consideration while defining the taxonomy of self-driving cars and the level of automation. The driver's attention needed for example, does the driver need to keep attention on the steering wheel all the time? The driver action needed, for example, does the driver need to steer? Does the driver need to control the speed? Or does the driver need to change the lanes or can the car stay in the current lane without any intervention? What exactly do we need to expect when we say that the car can drive autonomously? All these questions lead to the autonomous driving taxonomy.

The categorization standards that we will discuss in this topic are being suggested by the Society of Automotive Engineers (SAE), but we need to describe the driving task before classifying the levels of automation. The driving task consists of two main tasks, lateral control, and longitudinal control. Lateral control refers to steering and navigating laterally on the road, keeping a constant distance from the boundaries of the road. While longitudinal control is the task where we control the position and velocity of the car along the roadway, via throttle and brakes.

More tasks could be considered, like object and event detection and response (OEDR). OEDR is essentially the ability to detect objects and events that immediately affect the driving task and to react to them appropriately. Moreover, one more task to be considered is planning, which is primarily concerned with the long and short term plans needed to travel to a destination or execute maneuvers such as lane changes and intersection crossings. Some more miscellaneous tasks that people perform while driving can be considered as well. These include actions like signaling with indicators, interacting with other drivers, etc.

Levels of Automation

These levels are commonly used to describe levels of driving automation, defined by the SAE Standard J3016 [3].

Level 0 -- No Automation: It is a full human perception, planning, and control. At this level, there is no driving automation whatsoever, and everything is done by the driver.

Level 1 – Driving Assistance: In this level, the autonomous system assists the driver by performing either lateral or longitudinal control tasks, either but not both. For example, adaptive cruise control, in adaptive cruise control or ACC, the system can control the speed of the car, but it needs the driver to perform steering. So it can perform longitudinal control but needs the human to perform lateral control. Similarly, lane-keeping assist systems, in lane-keeping assistance, the system can help you stay within your lane and warn you when you are drifting towards the boundaries.

Level 2 – Partial Driving Automation: In this level, the system performs both the control tasks, lateral and longitudinal in specific driving scenarios. Some simple examples of level two features are GM Super Cruise and Nissan's Pro Pilot Assist. These can control both your lateral and longitudinal motion but the driver monitoring of the system is always required. Nowadays, many automotive manufacturers offer level two automation products including Mercedes, Audi, Tesla, and Hyundai.

Level 3 – Conditional Driving Automation: In this level, the system can perform Object and Event Detection in Response to a certain degree in addition to the control tasks. However, in the case of failure, the control must be taken up by the driver. An example of level three systems would be the Audi A Luxury Sedan, which was an automated driving system that can navigate unmonitored in slow traffic.

Level 4 – High Driving Automation: In this level, we arrive at highly automated vehicles, where the system is capable of reaching a minimum risk condition, in case the driver doesn't intervene in time for an emergency. Level four systems can handle emergencies on their own, but may still ask drivers to take over to avoid pulling over to the side of the road unnecessarily. With this amount of automation, the passengers can check their phone or watch a movie knowing that the system can handle emergencies and is capable of keeping the passengers safe. However, level four still permits self-driving systems with a limited operational design domain (ODD). For example, as of fall 2018, only Waymo has deployed vehicles for public transport with this level of autonomy. The Waymo fleet [4] can handle the driving task in a defined geographic area with a nominal set of operating conditions, without the need for a human driver.

Level 5 – Full Driving Automation: In this level, the system is fully autonomous and its ODD is unlimited. Meaning that it can operate under any condition necessary. Level five is the point where our society undergoes transformational change. With driverless taxis shuttling people in packages wherever we need them. Unfortunately, we don't have any examples for level five yet.

Objective

The source of the problem for us in a lot of driving situations is not our hands or feet but it's our eyes. We believe making a deep neural network that can focus on driving critical objects and ignore irrelevant elements like the background would improve the accuracy in steering angle prediction significantly.

Incorporating gaze info in self-driving cars is mostly used in driver assistance systems (ADAS). Only two papers published in late 2019 discussing the incorporation of gaze information into the end-to-end self-driving model [5][6].

Our idea is to get the most benefit from eye gaze by tracking it while driving. Adding Eye Gaze as an additional input to a deep neural network along with the scene taken from a front-facing camera is expected to generalize the model and predict more accurate steering commands in different driving environments without being explicitly trained in them. This will also help in decreasing the processing power by only processing a portion of the image.

Chapter 2

Background

Machine Learning

Machine Learning (ML) is considered a subset of artificial intelligence (AI) which enables the system to automatically learn from experience and deal with new problems and tasks effectively without being explicitly programmed. The existence of complex tasks in the real world which we can't handle with traditional rule-based programming accelerates the research in the area of machine learning to build a reliable system that can perform these complex tasks with high immunity to random possible variations. Machine Learning simply builds a mathematical model based on given information known as training data and use this model to perform predictions or decisions on relevant data that it hasn't been exposed to before.

History: In 1959, Arthur Samuel coined the term "Machine Learning" while at IBM and wrote the first computer learning program which was the game for checkers. As time passes, machine learning researches increased but considered only an application for artificial intelligence. In 1957, Frank Rosenblatt designed the first neural network for computers (the perceptron) simulating the process of a human brain. In the 1990s machine learning recognized as a separate field and started to flourish. In the 2000s with the huge computational technological advancements, more machine learning researches were done and machine learning becomes a trending topic in the research area.

Types of Machine Learning: The types of machine learning differ in their objective, inputs, outputs, and the approach to perform required tasks. We are going to cover the most used and essential types of machine learning briefly in the following subsections.

Supervised Learning: Supervised learning is the type of machine learning meant to map input data to output data. We build a machine learning model and train it using labeled data, it correlates the main features in the input data to the output labels and gains the ability to perform future predictions on relevant new unseen inputs with high accuracy.

Supervised Learning has many types and approaches. The most well-known types are regression and classification. Regression is used to predict continuous values of outputs depending on the current input to the model with the help of what we call hypothesis function. On the other hand, classification is being used to determine the category of a specific input. Classification could be binary (categorize input into two types only), or Multi-class Classification (categorize input into multiple options).

Unsupervised Learning: Unsupervised Learning is a type of machine learning which can learn by itself without the need for a labeled dataset. It searches for common correlations in given data, estimates a model that can analyze new and unseen input data. Unsupervised Learning is commonly used in clustering data into clusters that are determined without human interference based on the given unlabeled data [7].

Reinforcement Learning: Reinforcement learning is the type of machine learning where the agent learns by himself without any given data as it learns from interaction with the surrounding environment. Depending on the effects of specific actions the agent performs, feedback signals are sent to the agent to tell him how good/bad these actions were. Given the appropriate amount of time, the agent will be able to learn patterns and logical triggers to his actions so that the least amount of negative feedback will be sent to him.

Reinforcement learning, due to its generality, is studied in many other disciplines, such as game theory, control theory, and operations research.

Deep Learning

Deep Learning [8] is a class of machine learning which uses multiple layers to extract complex high dimensional features from raw input data. Deep learning can deal effectively with complex problems such as analysis of images, videos, and time-series events, taking into consideration spatial, temporal dependencies, or both. The term "deep" in deep learning refers to the number of layers through which the data are transformed. Deep learning methods can handle efficiently supervised learning problems, unsupervised learning problems as well as reinforcement learning problems.

Deep Learning Approaches: Deep Learning approaches are based mostly on artificial neural networks (ANN). Neural networks, in general, are built to simulate the behavior of the human brain– specifically, pattern recognition and the passage of input through various layers of simulated neural connections.

Neural networks are based on a collection of interconnected layers of nodes called perceptrons, responsible for processing information passing through different layers. Neural networks have many types based on the problem they are addressing. We are interested in the following three types;

Deep Neural Networks: A deep neural network (DNN) is a neural network with more than two hidden layers. As we increase the depth of the neural network, the ability to detect higher-level features increases. The main advantage of DNNs to traditional machine learning approaches is that we don't need to separately extract features from the raw input as the DNN can handle the task of feature extraction efficiently correlating the most affecting features to perform the required task. DNNs are trained using a backpropagation algorithm which is simply calculating the derivatives of a layer to the previous layer starting from the output layer to the input layer.

Convolutional Neural Networks: A convolutional neural network (CNN) is a class of deep neural networks, most commonly applied to analyzing visual imagery. CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually refer to fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data.

Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. However, CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extremity.

They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics [9][10]. Convolutional networks were inspired by biological processes [11][12][13][14] in that the connectivity pattern between neurons resembles the organization of the animal visual cortex.

Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field. CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage. They have applications in image and video recognition, recommender systems [15], image classification, medical image analysis, and natural language processing [16].

Recurrent Neural Networks: Recurrent neural network (RNN) is a class of neural networks that are used to analyze sequential data. The input to RNN is correlated with previous inputs forming a time series input that passes through the network affecting the final output as shown in figure 10. RNNs are suitable for problems with high temporal dependencies such as speech analysis, recognition, language models, machine translation, etc.

Deep Learning in Self-Driving Cars

The self-driving car is defined as a car that is capable of sensing and approximating its surrounding environment and navigating with little or no human interference. Deep learning had a major contribution in developing self-driving cars. Autonomous driving has two approaches to, either a hand-engineered modular pipelined approach or an end-to-end deep learning-based approach.

Modular Approach: The main idea of this approach is to split the task of autonomous driving into multiple modules performing smaller and specific tasks. Combing all these modules together gives the vehicle the ability to make decisions on its own without human interference. We are briefly discussing the main modules existing in this approach.

Localization: Localization means that the vehicle can detect its position with very high accuracy. HD maps are used for localization with the help of GPS.

Planning: Planning is meant to feed the vehicle with both the long term planning and short term planning. The planning module is important for the vehicle as it affects directly the behavior of the vehicle at every moment.

Perception: Perception module is the eyes for the vehicle. Several sensory data can be combined to provide a robust representation of the surrounding environment, like cameras, LIDAR, RADAR, and other sensors. CNNs are used in this module heavily to perform different tasks as lane detection, object detection and localization, and more.

Control: Given sensor data and planned trajectories, a control module is necessary to control the vehicle in a way that lets it follow its trajectory as well as interacting with the surrounding environment accurately.

End-to-End Approach: End-to-End approach aims to eliminate any hand-engineered pipelining, unleashing the abilities of deep learning to form its model of the environment, and an approximate robust relation between the surrounding environment and the corresponding control signals. The driving model learns from thousands of frames associated with control signals on how to deal in different situations without the need to program it explicitly. The resulting driving policy of this approach is a replica of the driver's behavior existing in the provided training dataset. End-to-End is also called imitation learning which is mimicking the behavior of the expert which in the case of self-driving cars is the human driver.

A question that is always being asked: how can a human teach a car to drive? Humans learn to drive by watching other people drive and then they try to observe, learn, replicate their actions, and get better. This is exactly what end-to-end imitation learning is about. The end-to-end system optimizes all processing steps simultaneously and eventually leads to better performance and a smaller system size. It has been applied in tasks such as road following and achieved great results by Nvidia [17], it was about using behavioral cloning with a human driver's with the camera frame input and observing the human behavior from the steering angle labeling.

A beginner human driver learning from an expert one, not only observes his steering actions but also he is observing his full-body actions, including his head and eye movement, his location of attention in different situations, his emotional statues, besides the verbal instructions which are given.

Among the human senses, the eye is the most one giving a huge amount of information to the brain, so we choose human gaze behavior as it contains a huge amount of information of the human brain complex attention mechanisms which will help to teach the model better, and make it recognize the most important frame components to focus on, which will help "humanizing" the self-driving cars.

Chapter 3
Dataset

3.1 Main Frame

For our project, we needed a dataset that has the driver's gaze position information for each frame along with the front-facing camera frame and the steering angle. A dataset that fulfills these requirements is very rare to find. Because eye tracking is not used much in divining applications. We had more than one candidate but they weren't specifically for our task and not accurate enough to do the job. We settled on the "DR(eye)VE dataset" [18]. "DR(eye)VE" is currently the largest publicly available dataset including gaze information and driving behavior in automotive settings. It consists of 74 video sequences of 5 minutes each of actual driving experience, for a total of 555,000 frames. Eight different drivers alternate during the recording process to smooth the bias given by each person's peculiar way of driving. Each video sequence is five minutes, covering different weather conditions (sunny, cloudy, and rainy), different lighting (morning, evening, and night) and different Scenarios (countryside, highway, and downtown). Videos were recorded with a roof-mounted camera of resolution 1920x1080 (RGB) and a frame rate of 25 fps, so each video contains 7500 frames.

Figure 1 Examples taken from a random sequence of DR(eye)VE. From left to right: frames from the eye-tracking glasses with gaze data, from the roof-mounted camera, temporal aggregated fixation maps, and overlays between frames and fixation maps.

As a first step, we focused on using the gaze position information in a simple lane following task with constant speed. As a result, the downtown videos' surroundings are out of the project's scope. The chosen videos are from the countryside and highway, as there are no people crossing roads at these videos and no crossroads and multi-paths. 12 videos were chosen for this project, which is about 90,000 frames (80% training, 20% testing).

Table 1 Chosen videos form DR(eye)VE dataset

Video Number	Lightning	Weather Condition	Scenario
01	Evening	Sunny	Countryside
02	Morning	Cloudy	Highway
03	Evening	Sunny	Highway
14	Morning	Rainy	Highway
20	Evening	Sunny	Countryside
22	Morning	Rainy	Countryside
37	Morning	Rainy	Highway
42	Evening	Cloudy	Highway
44	Morning	Rainy	Countryside
52	Evening	Sunny	Highway
56	Night	Rainy	Countryside
59	Morning	Cloudy	Highway

‘

3.2 Gaze Position Information

The driver's gaze information was captured using an accurate eye tracking device -commercial SMI ETG 2w Eye Tracking Glasses (ETG) shown in Fig 2. It tracks users' pupils at 60Hz and provides gaze information in terms of eye fixations and saccade movements. For each frame in the dataset, the driver's gaze information was acquired and registered to the external view recorded from a roof-mounted camera.

The dataset had a text file for each video contains the labels for each frame including course angle -which will be discussed later- and the gaze position of the driver. The gaze position is given as an x-y position in the mainframe. The gaze labeling of the dataset consists of 3 categories for the driver statues: saccade, fixation, and blink, which represent the movement of the eye gaze position and its different state.

Figure 2 SMI ETG 2w Eye Tracking Glasses

Chapter 4
Gaze Network

Predicting the driver's focus of attention -eye gaze- is essential for our work. What the driver is looking at is a personal behavior while what most drivers look at is a task-driven behavior that holds common gaze patterns shared among different drivers. This gives an intuition that by identifying the right factors affecting human's attention, the eye gaze can be predicted. A study showed that the semantic of the scene, the speed, and bottom-up features all influence the driver's gaze[19]. A paper published in 2018 introduced a computer vision model able to replicate the human attentional behavior during driving task[19]. They developed a deep learning model that can profitably learn to predict where a driver would be looking at in a specific situation using DR(eye)VE dataset and a multi-branch deep neural network.

This work argues that the act of driving combines complex attention mechanisms guided by the driver's past experience, short reactive times, and strong contextual constraints. Thus, very little information is needed to drive if guided by a strong focus of attention (FoA) on a limited set of targets and proposes a deep neural network (DNN) model that aims at predicting these targets.

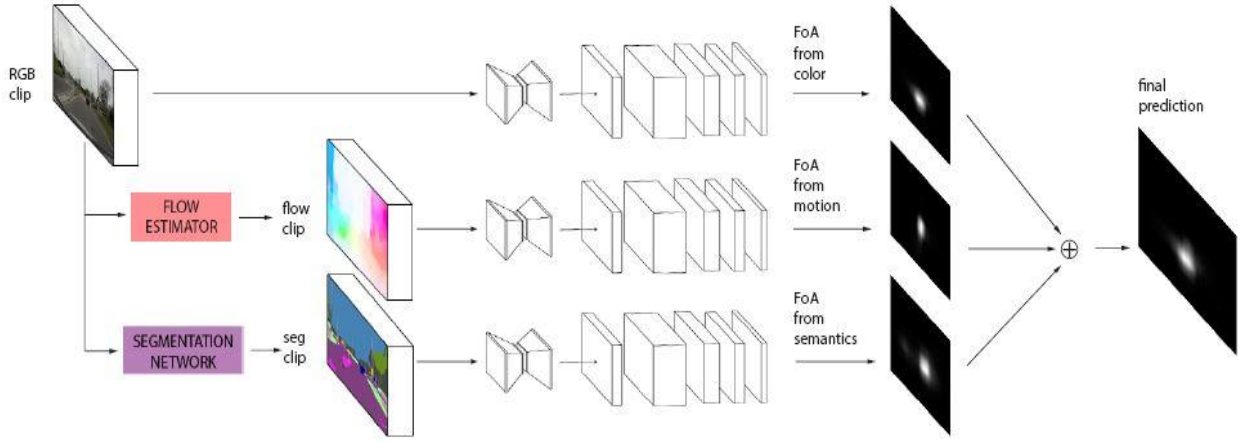

Figure 3 Multi-branch deep neural network for gaze prediction

As shown in Fig.3, The DNN is based on three different branches, each of which has its own set of parameters, and their predictions are summed to obtain the final map. The DNN estimates attentional maps from:

1- Visual information of the scene
2- Motion cues (in terms of optical flow)
3- Semantic segmentation

Each branch has been respectively fed with 16 frames clips in raw RGB color space, 16 frames clips with optical flow maps, encoded as color images through the flow field encoding, and 16 frames clips holding semantic segmentation from encoded as 19 scalar activation maps, one per segmentation class.

Figure 4 From the left : Main frame from roof mounted camera , Optical flow of the scene & Semantic segmentation of the scene

The COARSE module shown in Fig.5 is applied to both a cropped and a resized version of the input tensor, which is a video clip of 16 consecutive frames. The cropped input is used during training to augment the data and the variety of ground truth fixation maps. The prediction of the resized input is stacked with the last frame of the video clip and fed to a stack of convolutional layers (refinement module) to refine the prediction. Training is performed end-to-end and weights between COARSE modules are shared. At test time, only the refined predictions are used. Note that the complete model is composed of three of these branches, each of which predicting visual attention for different inputs (namely image, optical flow, and semantic segmentation).

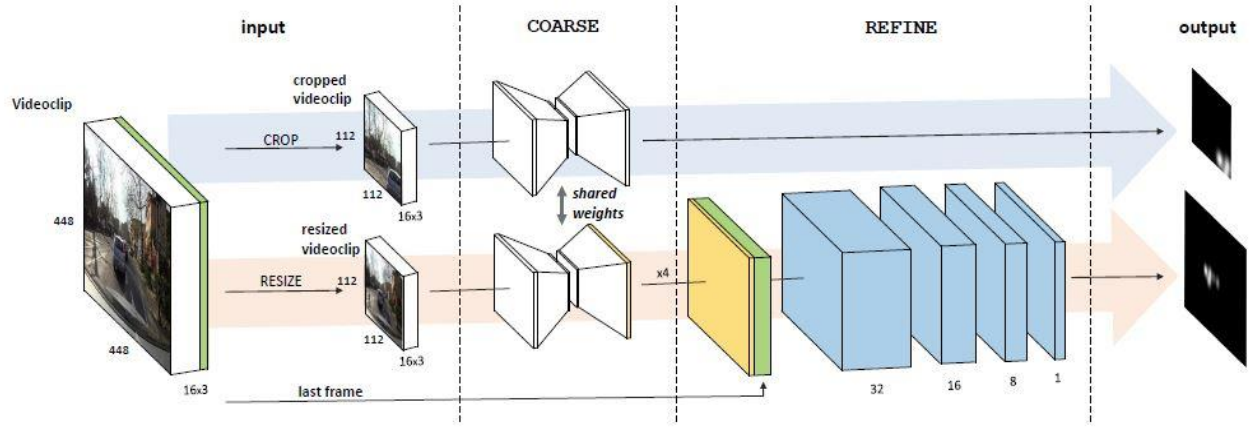

Figure 5 A single FoA branch of our prediction architecture

During the implementation of this network in our work, we faced several problems. To get the gaze maps from this multi-branch DNN, we had to get the inputs of each branch ready. We had to extract optical flow frames and sematic segmentation from the videos. the problem was that the authors didn't intend to make the network real-time or even fast enough for real-time applications. To get the frames for the semantic segmentation. The authors of the paper used "Semantic segmentation for dilated convolution"[20] to get semantic segmentation frames. The DNN described this paper was too large with 134 million parameters taking approximately 23 seconds on Google-Colab GPU to predict one frame, This was too long and would take weeks for us to just get the data ready for the gaze network and we had to search for an alternative. We used ICNET[21] DNN after modifying its output to be as close as possible to the output of the dilated convolution network in the same form that the main network accepts. The small difference between their outputs won't be significant in the final output of the Gaze network as the semantic segmentation branch doesn't contribute much in the summation of the 3 branches based on an ablation study conducted by the authors of the paper.

The speed of the IC-Net was much higher giving a prediction in 3 sec/frame, which decreased the semantic segmentation branch bottleneck and increased the speed of the whole model. The difference between the original network output and IC-Net output is shown in Fig.6

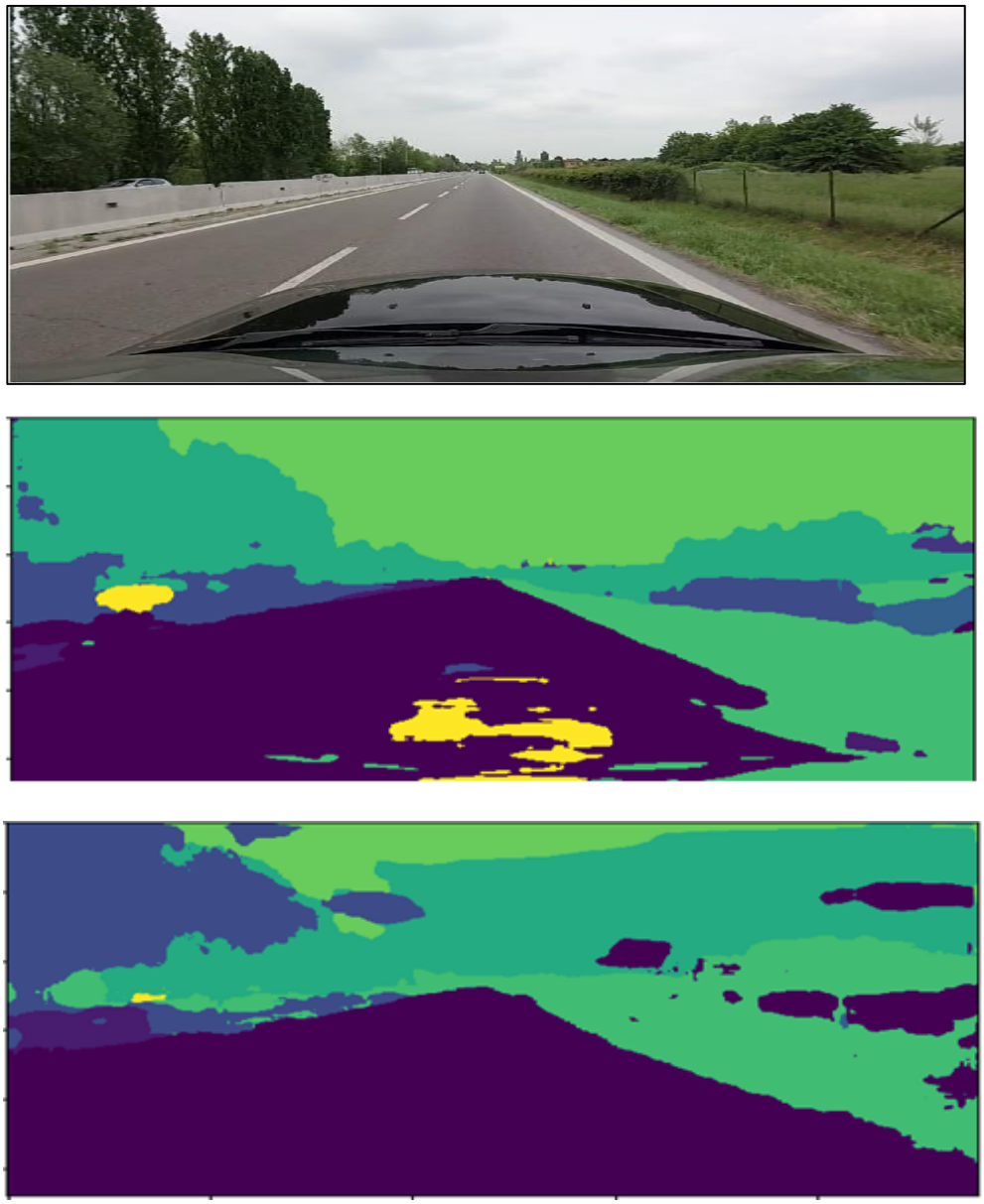

Figure 6 From the top : Main input frame, Dilation Conv-Net & IC-Net outputs

After getting all the inputs for the gaze network ready. We started the inference phase to get the gaze maps for our dataset. The execution time for the gaze net was too large, as it is a multi-branch DNN with 3 different branches and each branch takes 16 frames at once to process them. All of this resulted in an inference time of approximately 21 seconds on Google-Colab GPU to make one single prediction.

We thought of minimizing this time by getting the prediction of the 1st frame and the 6th frame and getting the frames in between using pixel-wise linear interpolation across the frames in time. This is based on the fact that the human gaze takes time to changes from one focus point to another, and the dataset is recorded at 25 fps which makes the consecutive frames almost the same with very small differences. Besides, the network takes 16 frames- clip every time to predict the FoA of each frame and shift these frames by one each time, so we have a huge quantity of redundant data. We measured the error in the frames predicted by the gaze net and the frames obtained using interpolation to verify our assumption. The error was below 1%. By using this approach, we could reduce the time greatly to about 7.8 % of the author's original setup inference time.

Then, we generated the gaze map prediction for each frame of our dataset to be used as the gaze information for our models. Shown in Fig.7 the final gaze map output of the Gaze Network.

Figure 7 from the left : Main frame & gaze map output of the Gaze Network overlayed on the main frame.

Chapter 5

Preprocessing

5.1 Frames Preprocessing

Now, we have the main front camera frame and the gaze information in the form of gaze map in a greyscale frame with sizes of 1920x1080 (RGB) for the mainframe, and 448x448 for the gaze map. We want to resize the 2 frames to be a size that is compatible with our baseline network which is PilotNet described by Nvidia[17]. PilotNet input size is 200x66 (RGB). As a result, we conducted the following preprocessing shown in Fig.8 to satisfy this requirement.

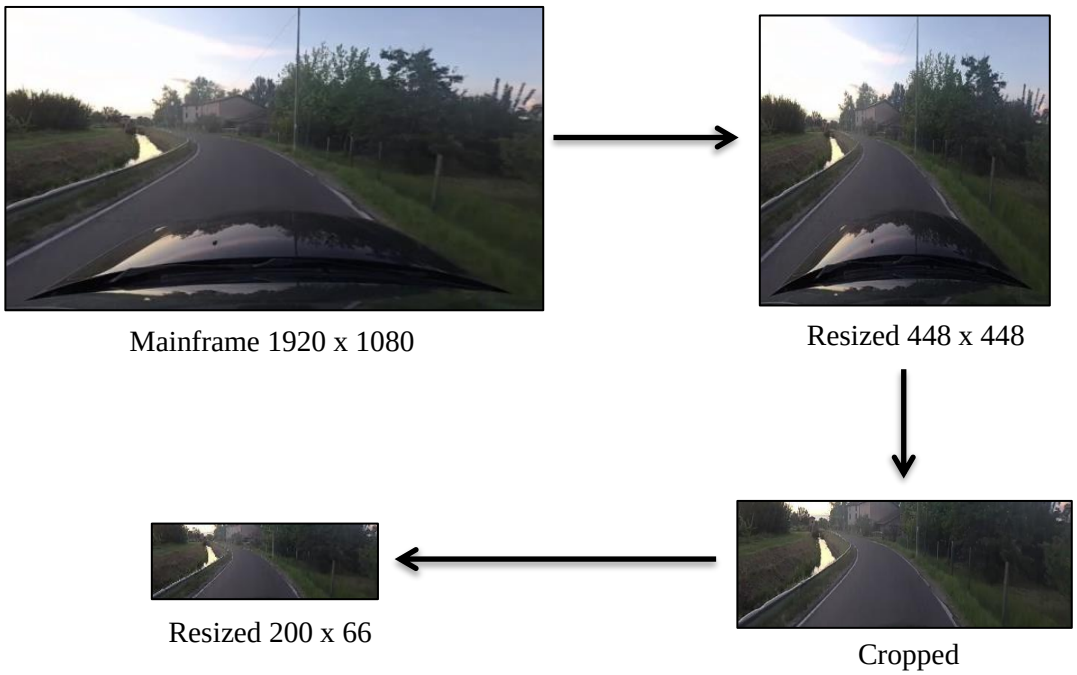

Figure 8 Preprocessing steps

I. Resizing

As we discussed in the Gaze Network chapter, the gaze network output the gaze map frame in the size of 448x448. The mainframe is resized to be the same size as the gaze frame as we are going to crop the mainframe to remove the unimportant parts. So, we want both of them to have the same region of space to not confuse the model, so its new size will be 448x448 (RGB) as shown in Fig.9.

Figure 9 Resized mainframe and gaze map frame (448 x 448)

II. Cropping

We used a cropping function that takes 4-numbers (Top, Bottom, Left, and Right) as input. These numbers are the edges for the required part from the frame to be saved. Because all videos are not recorded using the same car, and with different camera positions, the mainframe contains car front at various positions across the videos as shown in Fig.10, so we had to crop each video with a different value manually to compensate for these differences.

Figure 10 Differences between camera configuration across s frames from sample videos

We took a sample frame from each video then defined the lower bound of the cropped area to be the car front's upper bound, the following table shows lower bound to apply while cropping each video from our dataset:

Video #	**01**	**02**	**03**	**14**	**20**	**22**	**37**	**42**	**44**	**52**	**56**	**59**
Lower bound	303	312	364	309	363	338	305	303	338	341	306	295

Table 2 cropping area's lower bound for each video

Also, we cropped part from the sky, as it contains no useful information for the network. As a result, for all videos Top=140, Left and Right are not cropped so Left=0 and Right=448. This cropping would be applied on both, mainframe and gaze frame to have the same information at the same position mapped from mainframe to gaze frame.

Figure 11 Cropped main frame (448 x 448) (RGB)

III. Final Resizing

Now we have all frames empty from non-useful data but with different sizes, so at this stage we resized all frames -main and gaze frames- to be 200x66 which is suitable for our network.

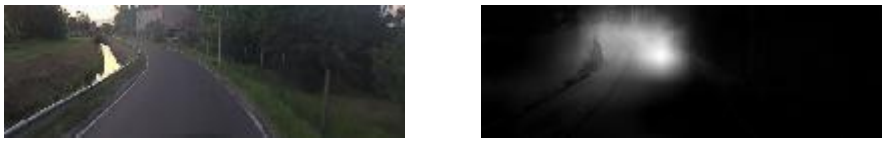

Figure6. Resized main and gaze map frames (200 x 66)

5.2 Course-Steering Angle Conversion Algorithm

After choosing our dataset, there was only one drawback, that this dataset wasn't labeled with steering angles. But instead, it had a course angle. The course angle is the angle between the head of the car and the North (Angle to the north). It can be seen as the angle made by a compass as the car is moving as shown in Fig.12

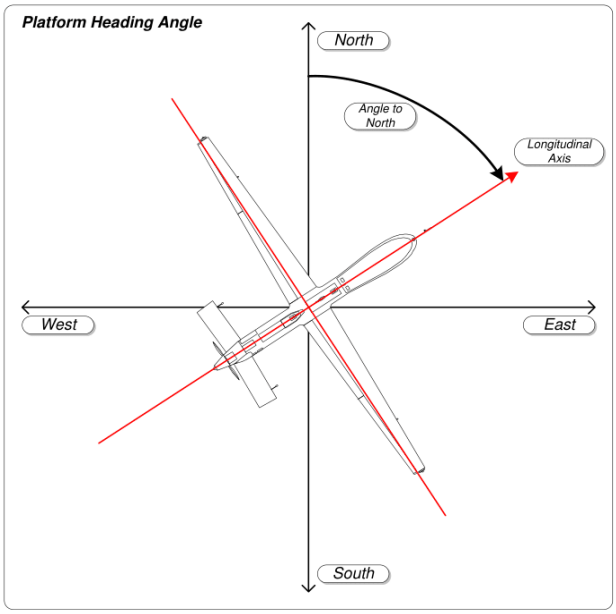

Figure 12 Course angle

As seen in Fig.12, you can guess that there is a relation between this angle and the steering angle, but we didn't find a direct simple relation without going deep into the mechanics of the car online. We managed to find this relation, as the dataset has no value to us without the steering angle labels with the frames and gaze maps. The initial idea to get the steering angle from this course angle was to use a derivative, as we can see that the relation between steering and course angle is accumulative. To get an intuition, if you were going straight heading to the north, so the steering and the course angle will be zero. Now suppose that the street is turning 90 degrees to the right, and you want to follow it. You will make the steering angle 90 degrees and as time passes, the course angle will be accumulated and continuously increasing as the head of the car is turning right until it reaches 90 degrees. You will get the steering wheel back making the steering angle zero, but the course angle will stay 90, so it's a relation of integration from the course angle point of view or differentiation from the steering angle point of view.

Figure 13 Initial idea block diagram

As shown in Fig.13, a discrete differentiator is applied to the course angle by subtracting every two consecutive angles. We apply an exponential Moving average to smooth the output and apply an averaging filter after that for more smoothing and better output. The initial idea produced good results and validated the idea, but wasn't good enough. As a result, we had to get the relation proved from the geometrical basis of and the motion equations to get a more accurate model.

To start from the beginning, we know that a car with a certain constant steering angle will move on a circular path as shown in Fig.14. Each one of the front wheels will move on a circle with a different radius and each wheel will have a slightly different angle to get a smooth motion on the circular path.

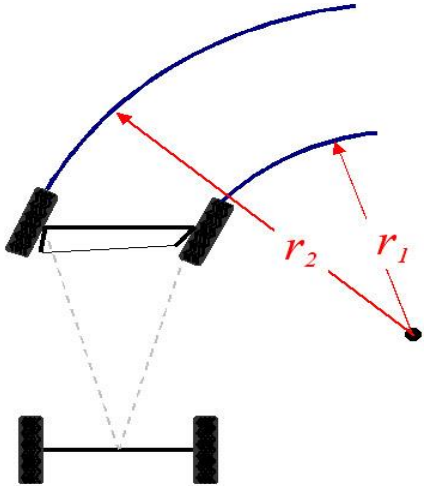

Figure 14 Circular path motion

Now let's have a look at the geometry of the steering angle of the car as shown in Fig.15.

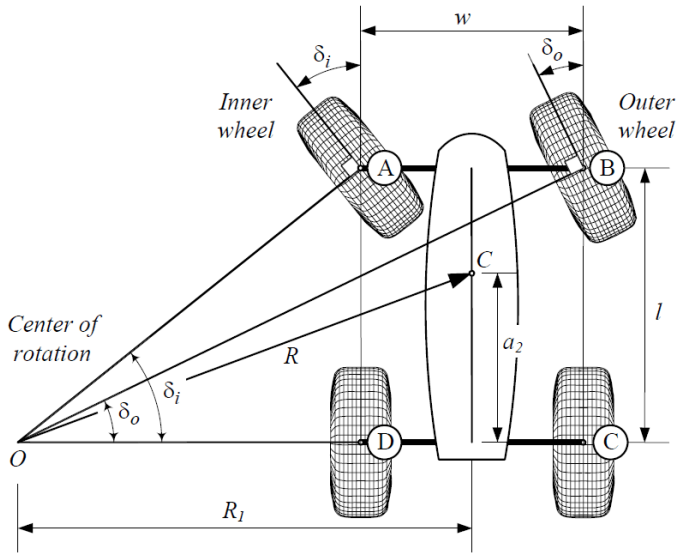

Figure 15 Car's steering geometry

We know that each one of the front wheels moves on a circle with a different angle. For simplicity of the model and without losing generality, we are going to use only the circle with the small radius "the one drawn by the wheel nearer to the center of the turning circle". The point "O" is the intersection of two lines drawn perpendicular to the front and the back wheels. It's the center of the motion circle. Its position will change with each steering angle changing the center and the radius of the circle. AD side length is ***l*** which is the "wheelbase length". It's the distance between the from and the back wheels and it differs from one car to another. From the trigonometry of the right-angle-triangle ODA with the "left wheel steering angle" = δ_i = ***SA***,

$$SA = tan^{-1}\frac{L}{R} \qquad (1)$$

We viewed the problem in a time instance "static form", now we move to the dynamic form as the car is moving.

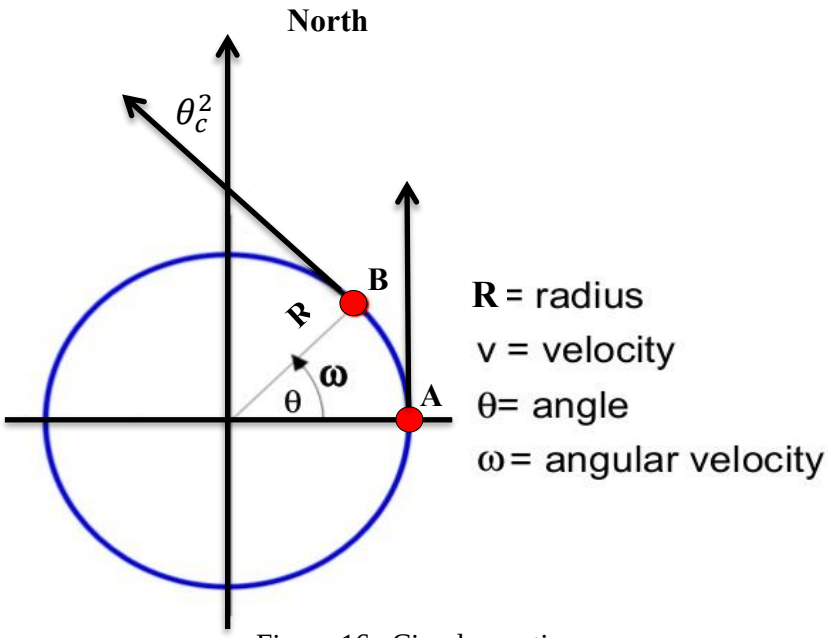

Figure 16 Circular motion

In Fig.16, we assume the car was at point A at a time t_o and the course angle, in this case, will be θ_c^1 = Zero. After time t with a constant steering angle SA the car will move on the arc AB and reach point B and the course angle will be θ_c^2. From the circular motion equations:

$$v = \frac{2\pi R}{t} \qquad (2)$$

From the geometry, we can conclude that $\theta = \theta_c$, so:

$$\theta_c^1 - \theta_c^2 = \frac{t*v}{R} \quad (3)$$

From (1) & (3) we conclude the final relation between steering and course angles:

$$\theta_S = SR * Tan^{-1}(\frac{(\theta_c^1 - \theta_c^2)*L}{v*t}) \quad (4)$$

θ_S : steering wheel angle
t: frame rate = 1/25
v: velocity at each frame
L: wheelbase length ≈ 2.6
SR: steering ratio ≈17
θ_c: Course angle

SR indicates the steering ratio, and it's the ratio of the steering wheel angle to the ratio of the car wheels steering angle. Since we don't know the type of car used in the dataset collection, we will use the average number for all the constants of the car type in the previous equation for steering ratio and the wheelbase length. For time ***t***, it will be the time between two frames of the video (frame rate 25).

Velocity at each frame is given in the dataset, so we now have strong mathematical proof to our conversion algorithm which considers velocity and car constants. The results of the previous model were very good and way better than the initial idea results. We can see know with mathematical proof the source of the differentiation from Equation (3).

Before applying the conversion equation to the course angle, we need to analyze it and apply some preprocessing to correct some problems that would appear after conversion.

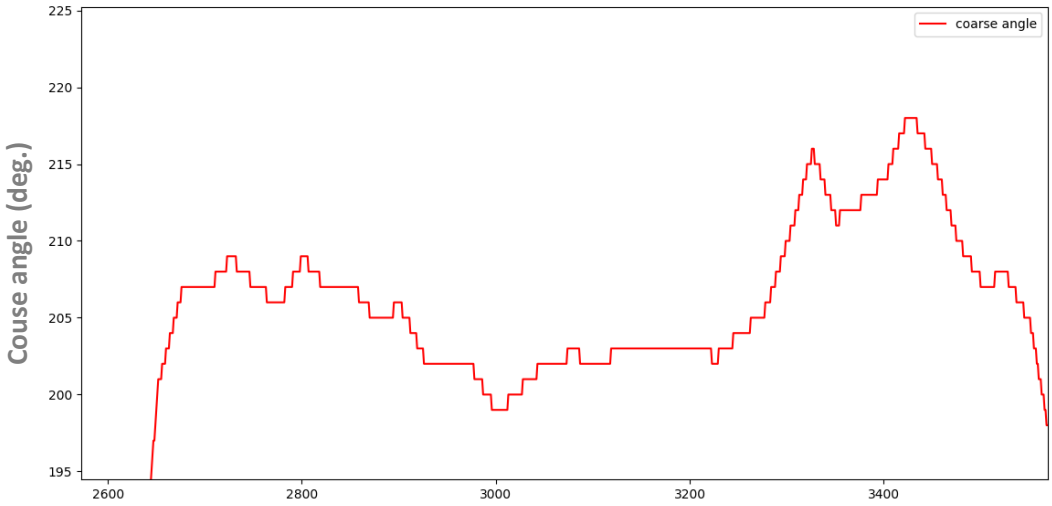

Figure 17 course angle sensor output

In Fig.17, we can see that the course sensor they are using is a discrete sensor, i.e. it changes with a step of one degree. This step is very small tending to zero with respect to the constant time before any sudden change, and this will cause the steering angle to go to infinity (very high value) when applying the equation as it contains derivatives. To solve this problem, the input data had to be smoothed, using an averaging filter which will cause a time delay and it's very sensitive to cause a delay in the steering angle, so we used a First Order Hold (FOH) to smooth the sharp edges and make the change happen on a larger number of frames as shown in Fig.18

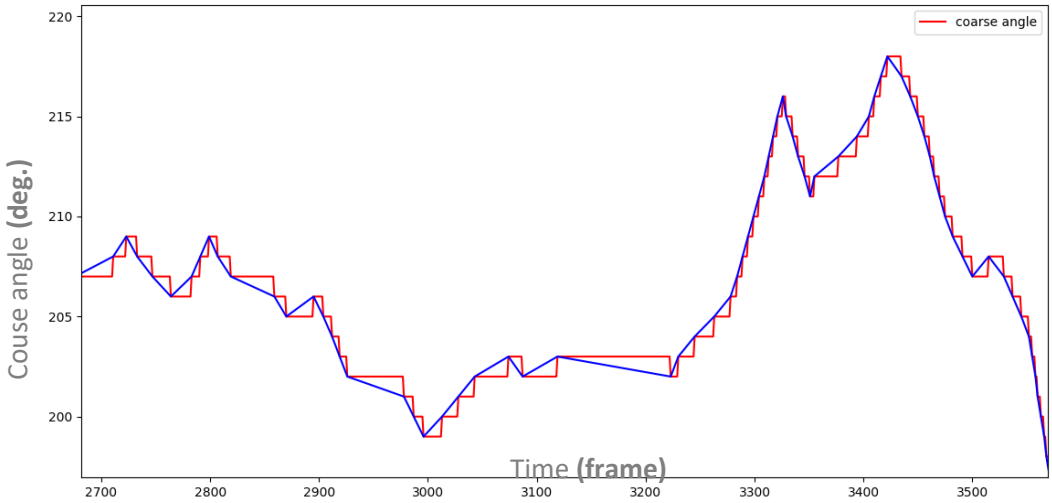

Figure 18 Smoothed Course angle

After applying equation (4), the steering angle was generated but with high oscillations, as the changing time of the course angle is relatively small and has discrete values at each time instant. We used the exponential moving average (EMA) to smooth out the curve without much time delay, giving the new steering angles a higher weight compared to the old one. The final output is shown in Fig.19.

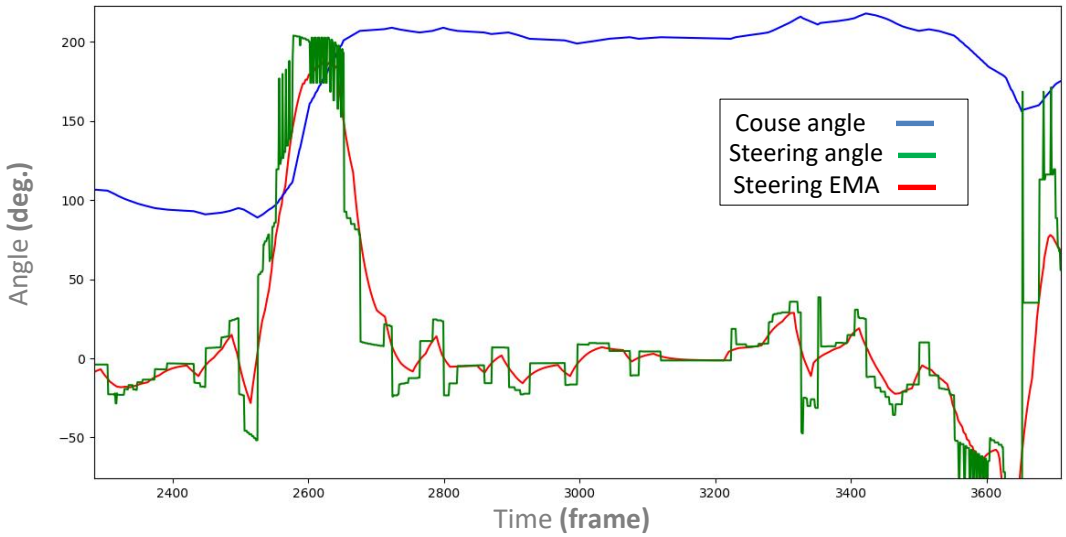

Figure 19 Smoothed steering angle

The last problem faced us in the dataset course angle conversion, is that some frames have the course angle changing from 0 to 360 or from 360 to 0 at a moment. As the car is changing its heading around the north direction, which makes the conversion fails as shown in Fig. 20

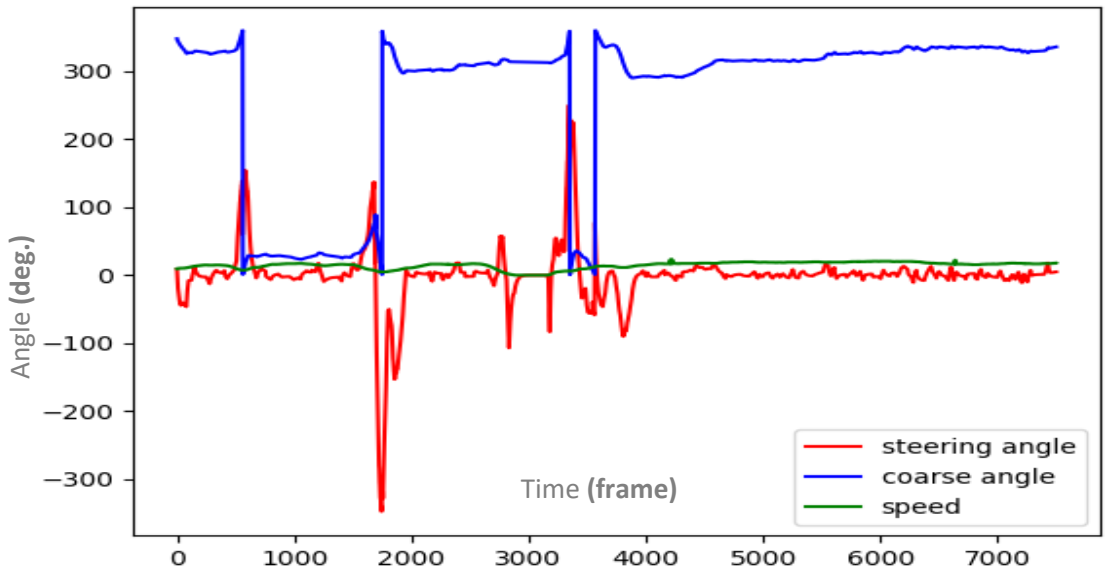

Figure 20 Course angle discontinuities

To solve this problem, a threshold was put and the angles that surpass this threshold will be subtracted from 360. This will make the continuity of the curve restored as shown in Fig 21.

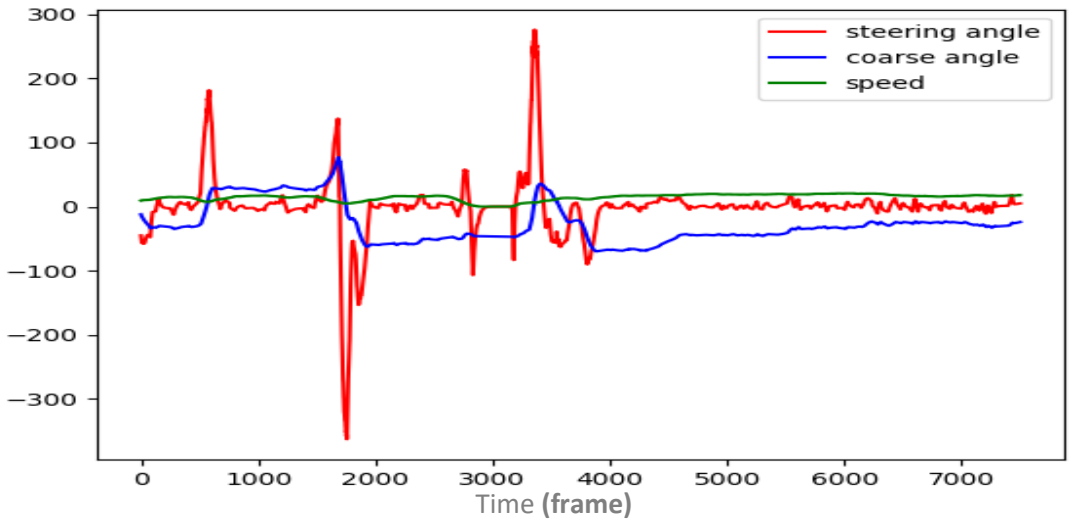

Figure 21 Final steering output

Chapter 6
Main Architectures

6.1 Baseline Architecture (PilotNet)

Driving in every country is different. There are different rules on the road, sometimes it’s very dramatic; such as driving on the other side. Sometimes, it is a little bit more subtle; such as rules about right turns on red lights. But the reality is self-driving cars that are driving in different cities have to have different kinds of brains behind them. They have to have different rules governing their behavior. So, the question is how can we develop a technology that we can scale and adapt to different cities around the world?

One of the methods to do that is using End-to-End learning. End-to-end models learn all the features that can occur between the original inputs (x) and the final outputs (y). This enables the computer to form a model by observing how humans drive in practice. Recently, Nvidia described PilotNet which is CNN that goes beyond pattern recognition[17]. PilotNet architecture is shown in Fig.22. It learns the entire processing pipeline needed to steer an automobile. Taking images from a front-facing camera as input and predicting the steering commands as its final output.

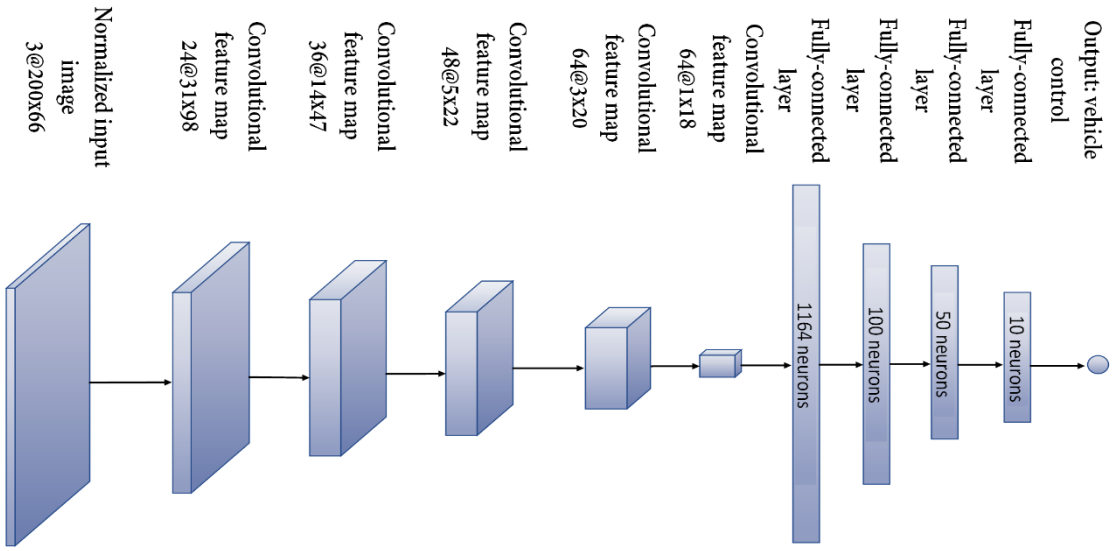

Figure 22 PilotNet Architecture

In our work, as there is no similar work to ours using the same dataset to compare the results, we used PilotNet as a baseline for our introduced models to investigate the improvement of incorporating eye gaze into the training process. We benchmark the root mean squared error (RMSE) values achieved by different architectures against PilotNet RMSE to measure the improvement percentage achieved.

6.1.1 Training

We trained PilotNet on our dataset using only RGB images as an input without any gaze information. The following hyperparameters were used to achieve the minimum RMSE value for our test set:

Optimizer: ADAM Optimizer
Learning rate: 0.0001
L2 Regularization constant: 0.00005
Drop out ratio: 0.2
Epochs: 40

After running 40 epochs, learning rate was decreased to 0.0001x0.5 for 15 more epochs to give RMSE = 0.0096080. Then decreased again to 0.0001x0.25 for more 35 epochs giving RMSE = 0.00716854.

*All the experiments were conducted on Google-Colab GPU

6.1.2 Results Analysis

PilotNet was able to score an RMSE value of **0.0105**.

To have some insights on how the model takes its decision to predict a steering angle command to make sure that the model is well trained and understand our dataset well, we implemented saliency maps using a method described in [22] to see what regions in the image are the most important for the network during steering angle decision making. As shown in Fig.23, It is obvious that lanes and both sides of the road are the most important features in the input image for the model, while the sides of the road and the background less contribute to the decision.

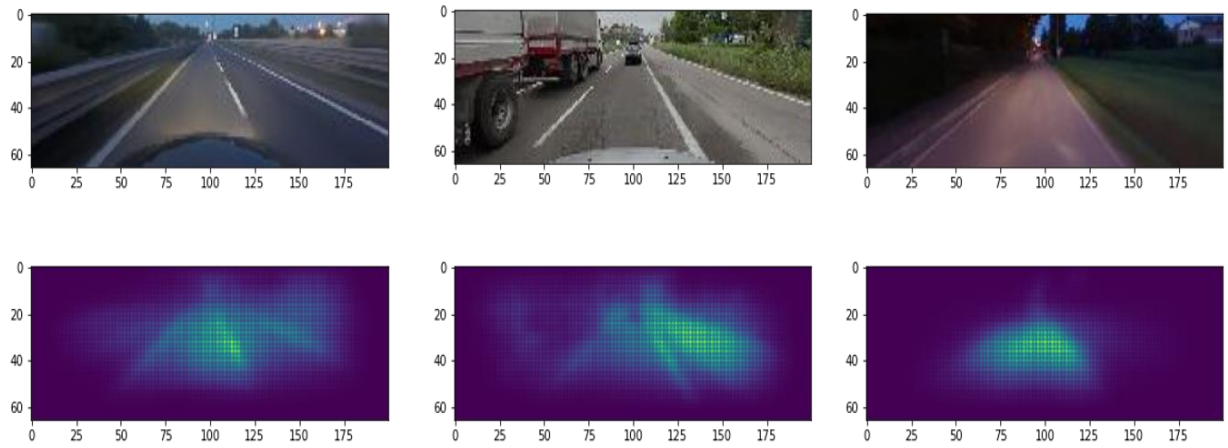

Figure 23 Saliency maps of PilotNet

\

6.1.3 Initial Gaze Incorporation Experiment

To get a sense of whether incorporating eye-gaze information in the training process will help in predicting a more accurate steering angle or not, we tried different combinations of stacking the gaze information with the RBG frame as an input to PilotNet and monitored the change in the RMSE scores.
We tried the following stacking approaches (shown in Table 3):

- RGB frame alongside with gaze map (4D input: 3D frame + 1D gaze map)
- RGB frame alongside with frame masked by gaze map (6D input: 3D frame + 3D frame masked by gaze map)
- RGB frame masked by the gaze map (3D input; as masking does not increase the dimensions of the RGB frame)
- RGB frame alongside with RBG frame masked by gaze heat map (6D input: 3D frame + 3D frame masked by heatmap)
- RBG frame masked by gaze heat map (3D input: 3D frame masked by heatmap)

The results achieved are summarized in table 1. We notice that all architectures achieve lower RMSE values than our baseline -which is PilotNet trained without gaze information-. This supports our idea that eye gaze improves steering angle prediction accuracy significantly. However, the best RMSE score is achieved by incorporating RGB frames masked by a gaze heat map, we think that this method has the advantage of eliminating repeated information in the input which may confuse the network.

Table 3 RMSE & Improvement results

Input	RMSE	Improvement (%)
Baseline	0.0105	-
RGB + RGB masked by gaze map (6D)	0.0098	6.6%
RGB + Gaze map (4D)	0.007793	25%
RGB masked by gaze map (3D)	0.00968	7.8%
RGB + Gaze heatmap (6D)	0.00898	14.5%
RGB masked by heatmap (3D)	0.00839	20%

Figure 24 Top left: RGB frame, Top right :1D gaze map, Bottom left: RGB masked by gaze heatmap & Bottom right: RGB frame masked by gaze map

6.2 Fusion Architectures

Fusion techniques include middle and late fusion. In fusion networks, we have multiple inputs. We try to extract features from each input then fuse these extracted features to make a prediction. The point at which fusion takes place defines the type of fusion. If the fusion occurs directly after extracting features then it is a middle fusion and if fusion between features occurs after passing the extracted features through more than one fully connected layer, it is called late fusion.

We tried middle fusion to give the network a chance to extract features from the input. Since we have two inputs; the RGB frame and the gaze map, our network consisted of two branches as shown in Fig. 25. We started by using the convolution layers (5 layers) in PilotNet as feature extractors from both the RGB frame and the gaze map. Afterward, we fuse the extracted features to one feature vector and go through a series of fully-connected layers ending with our final prediction.

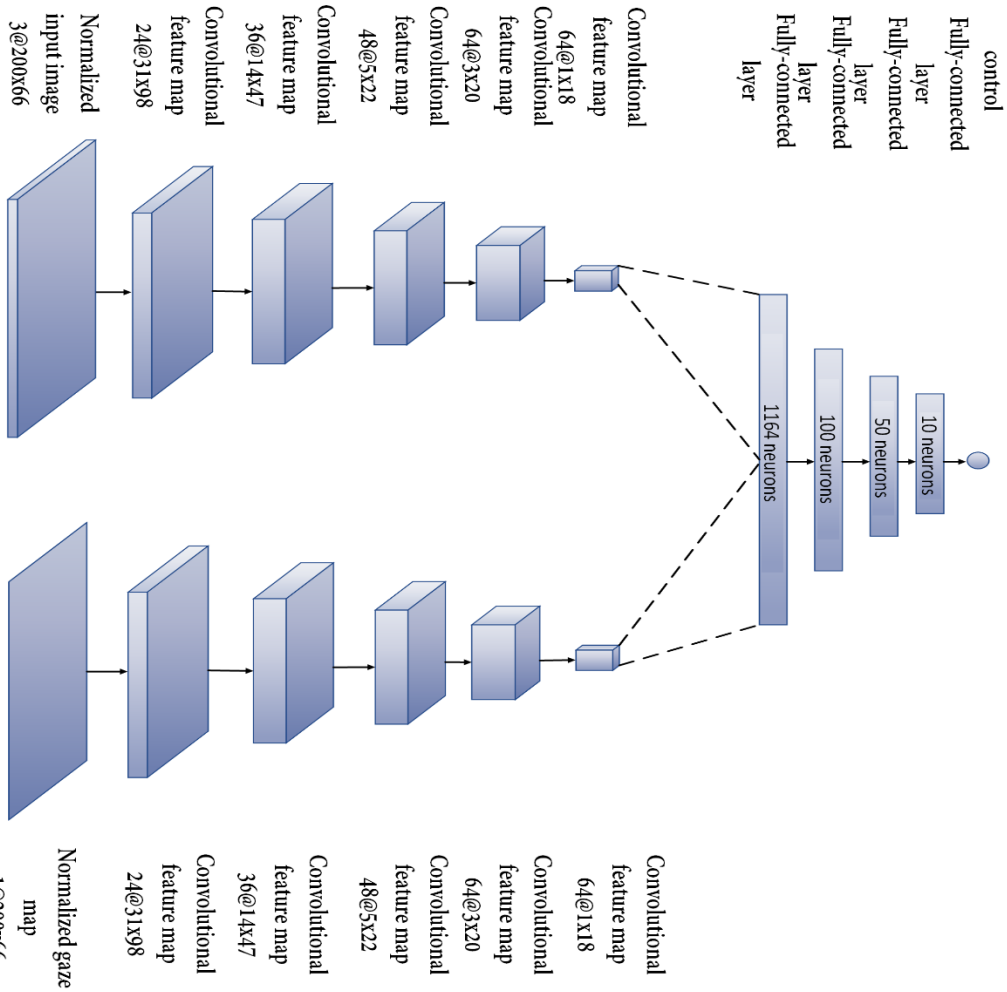

Figure 25 Middle fusion architecture

6.2.1 Results Analysis

The architecture in Fig. 25 had an RMSE value of 0.00963 and achieved an improvement percentage of 8.28 %**.** This architecture has 5180579 parameters**.** However, using the same number of convolutional layers as a feature extractor for both the gaze and the RGB frame is not fair. Because the gaze map is 1D and has much fewer features than the RGB frame which is 3D. Decreasing the number of layers in the gaze branch will decrease the number of parameters of the network and reduce its inference time.

As a result, we started trying out different experiments involved decreasing the number of convolutional layers in the gaze branch and changing the number of filters in each layer. Keeping in mind that the length of the fully-connected layers is the most contributing factor in the network's number of parameters. So, our main goal was to decrease the number of features entering the first fully-connected layer without hurting the network's performance. The final architecture is shown in Fig.26 which achieved an RMSE value of 0.00730 and an improvement percentage of 30.5%. The number of parameters in this architecture is 926,665.

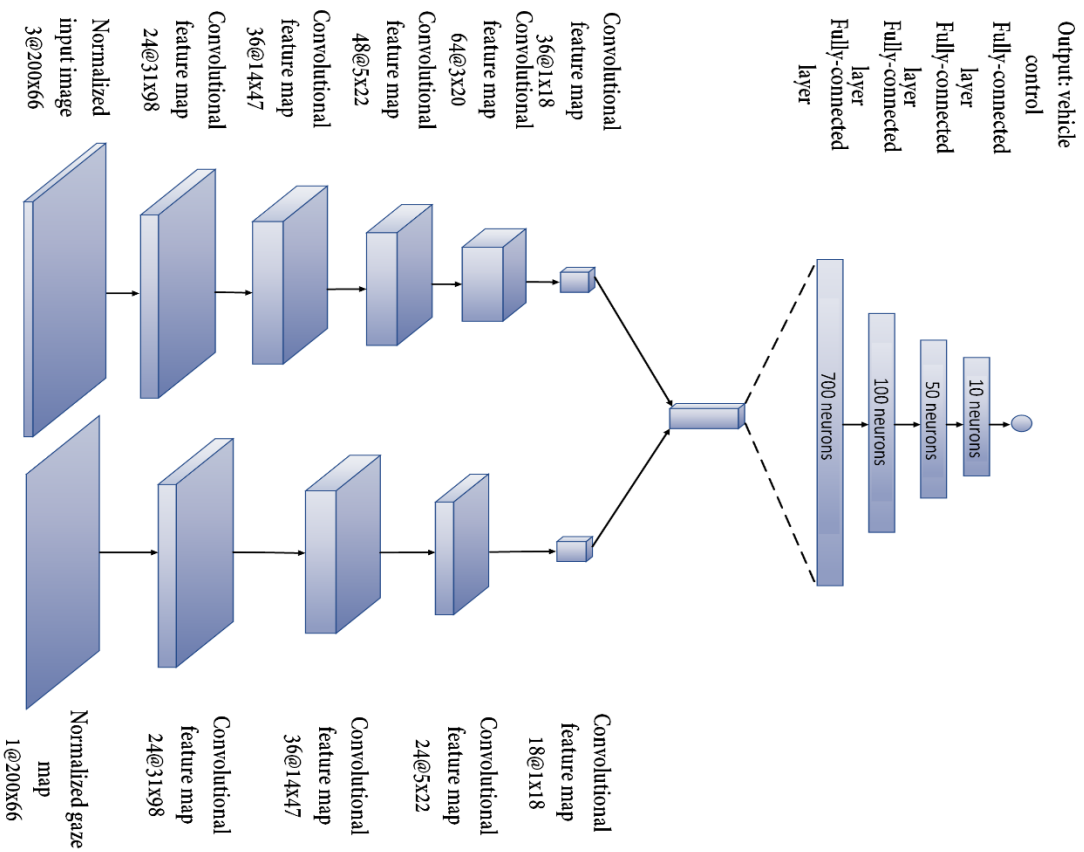

Figure 26 Middle fusion final architecture

As part of verifying the idea that incorporating the gaze information does improve the performance of the network, we tried to feed the network with an RGB frame in both of the branches. Technically, this does not make sense, however, we wanted to make sure that the improvement in the performance is due to the presence of the gaze maps not due to the increase in the size of the network. We followed the same hyperparameters as described above and the network achieve an RMSE value of 0.0082609.

We implemented saliency maps using a method described in [22] to see what regions in the image are the most important for the network during steering angle decision making. We calculated saliency maps with respect to both branches. In other words, we calculated saliency maps from the output back to the RGB frame input and back to the gaze map input.

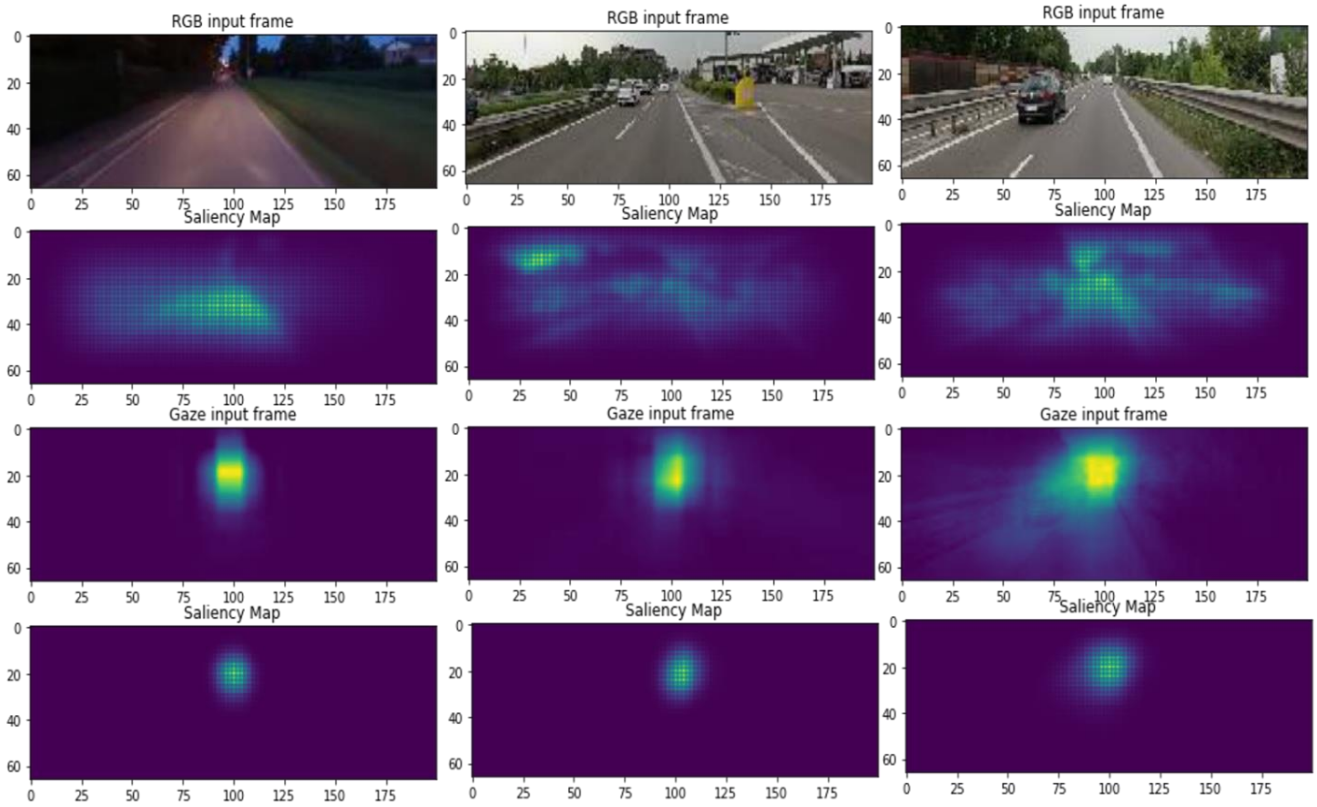

Figure 27 Middle fusion saliency maps

6.3 Spatial Transformer Networks (STN) Architecture

6.3.1 Neural-Attention Introduction

Informally, a neural attention mechanism equips a neural network with the ability to focus on a subset of its inputs or features. It expands capabilities of neural networks: they allow approximating more complicated functions, or in more intuitive terms, they enable focusing on specific parts of the input. The attention mechanism is newly introduced to the world of the neural network with more focus on topics like natural language processing and generating image captions [23][24]. Neural attention is very famous in fields like natural language processing as the attention concept was introduced first in that field, and it achieved very impressive success, after that, it was introduced to other fields of deep neural networks and also a big success is being achieved right now.

We thought of bringing that concept to our problem as its very related conceptually. The driver’s gaze position and information are directly correlated to the human brain's attention mechanisms to focus his gaze on a specific part of the image during driving. We wanted to simulate that optimized human behavior giving the SDC a strong ability to see the frames more efficiently.

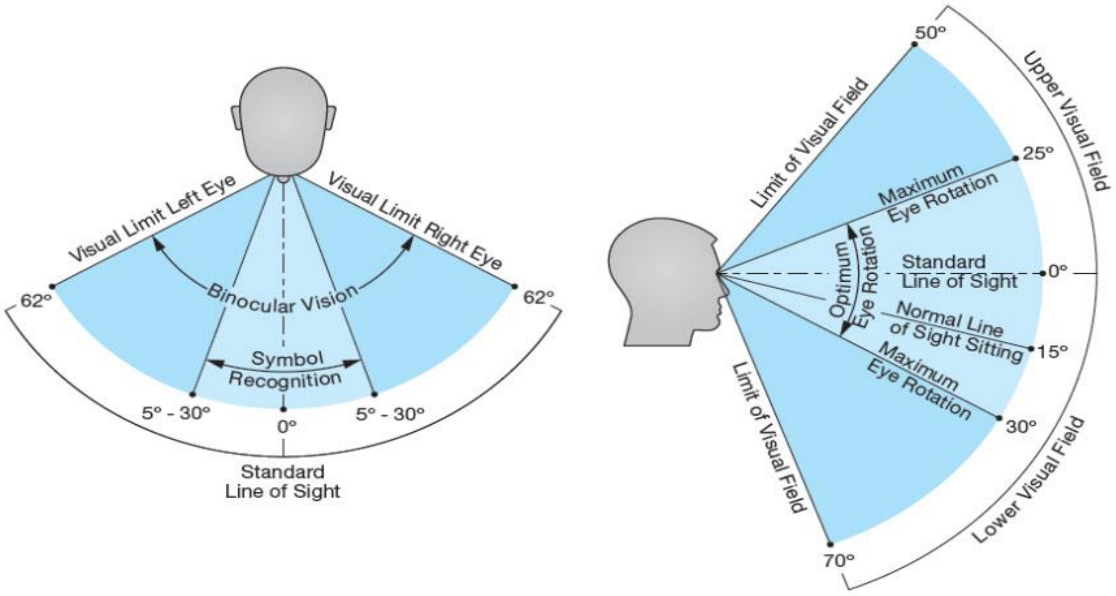

Figure 28 human visual field

As we can see in Fig.28, the human visual field from the top view is nearly 124 degrees. Not all that area have the same resolution or the same density of information that is feed to the brain. The human visual mechanism is very simple and very efficient, as most of the field collects information about the space but only about 8.3% of it has the highest information intensity with a high-resolution receptive field of about 20 degrees from a total of 124 degrees. That is why the human has to keep moving his eyes and adjust his gaze position in the area the brain decided is the most important at each time instant. Using this idea, a lot of attention mechanisms for computer vision were introduced like: soft attention, hard attention, and many more ideas and techniques. We will focus on the one used in this project which is “Spatial Transformer Networks”.

6.3.2 Spatial Transformer Networks (STN)

STN was introduced by Google Deep Mind in 2016 [25], it was Mainly directed to image classification problems. The idea was that Convolutional Neural Networks define an exceptionally powerful class of models, but are still limited by the lack of ability to be spatially invariant to the input data in a computationally and parameter efficient manner. The main idea of the paper was to allow for much more general transformation that just differentiable image-cropping.

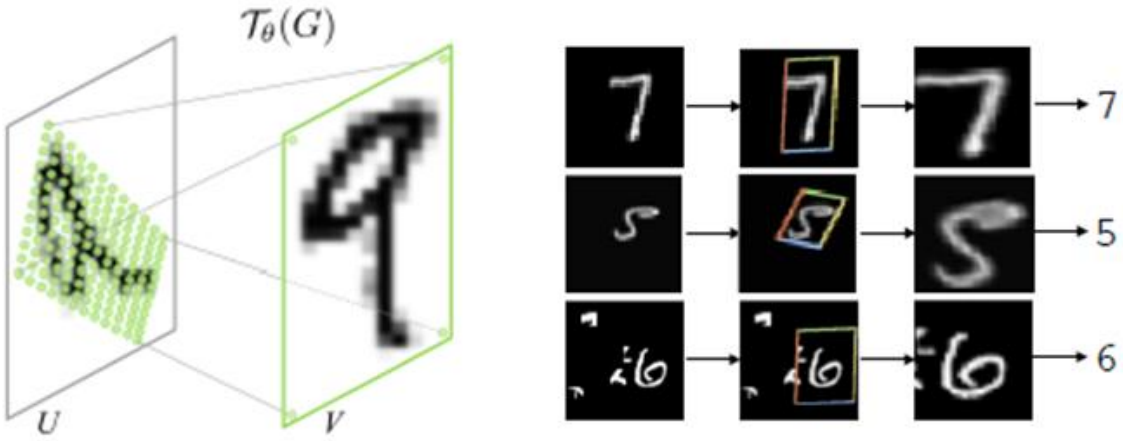

Figure 29 STN visulization

As we can see in Fig.28, the STN layer transforms the input frame so it's better recognized by the classifier. It can be added to any network without changing it and can be trained using backpropagation techniques as it is differentiable and this is its main strength.

STN consists of 3 main parts: Localization net, Grid generator, and sampler, as shown in Fig.30

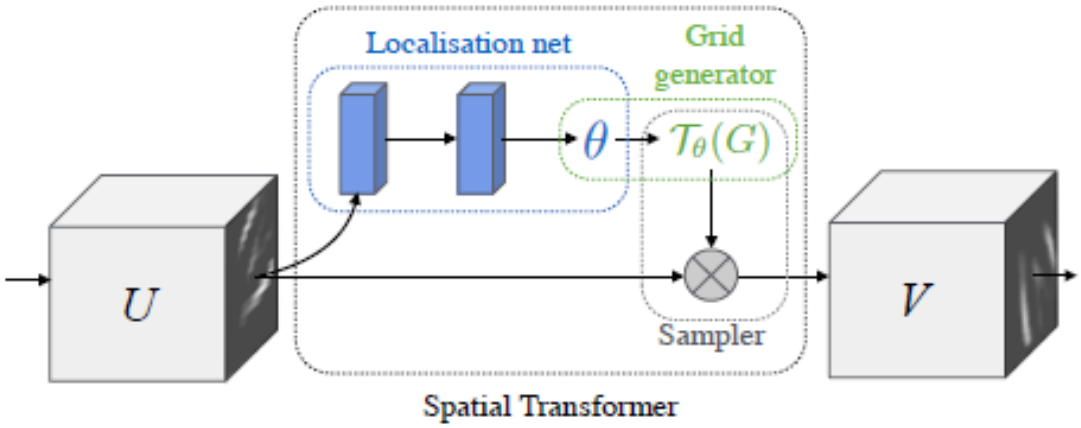

Figure 30 STN Components

The Localization net can be a CNN followed by a fully-connected (FC) or can be only FC as described by the paper. It takes the input feature map with width ***W***, height ***H,*** and ***C*** channels and outputs the parameters of the transformation ***T*** to be applied to the feature map.

$$\Theta = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix}$$

The 6 elements of localization net output are used by the transformer to do this transformation: **cropping, translation, rotation, scale, and skew.** All the transformation equations can be found in the paper. For the nature of our problem, we will only use only 3 transformations of the above 5: **cropping, translation, and isotropic scaling.** Any other transformation will confuse our steering angle generator network. So our theta matrix will be reduced to only 3 elements:

$$\mathtt{A}_\theta = \begin{bmatrix} s & 0 & t_x \\ 0 & s & t_y \end{bmatrix}$$

S parameter is for scaling, t_x, and t_y for shifting in x and y directions. Our architecture design of the Localization Network is shown in Fig.31. It consists of alternating convolutional and max-pooling layers to focus and locate the important features only, followed by 2 FC layers and a final FC layer with tanh activation function to get the theta 3 transformation parameters.

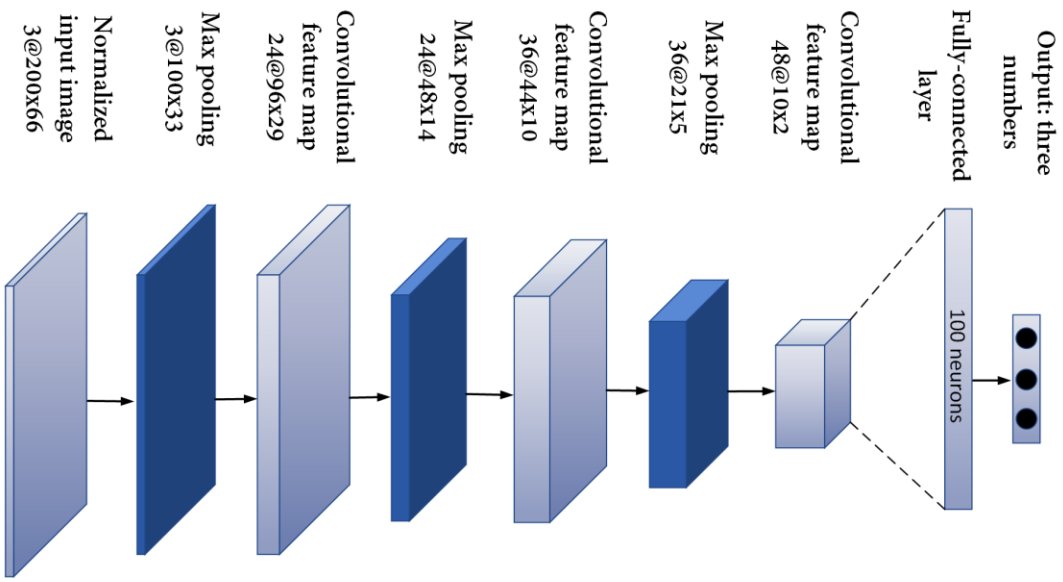

Figure 31 LocalizationNet architecture

Now we modify the connections the structure of the STN to fit our model, instead of feeding the mainframe to the localization network, we feed the mainframe overlaid with the gaze map as RGB heat map, so the localization network will learn to pick the region of interest from taking into consideration the information in the mainframe plus the information in the gaze map. Now we construct the final architecture as shown in Fig.32. The input frame and the gaze map will be the inputs for the STN layer, the output will be the mainframe with the STN transformation applied on it which will be fed to Nvidia PilotNet. The whole architecture will be trained end to end.

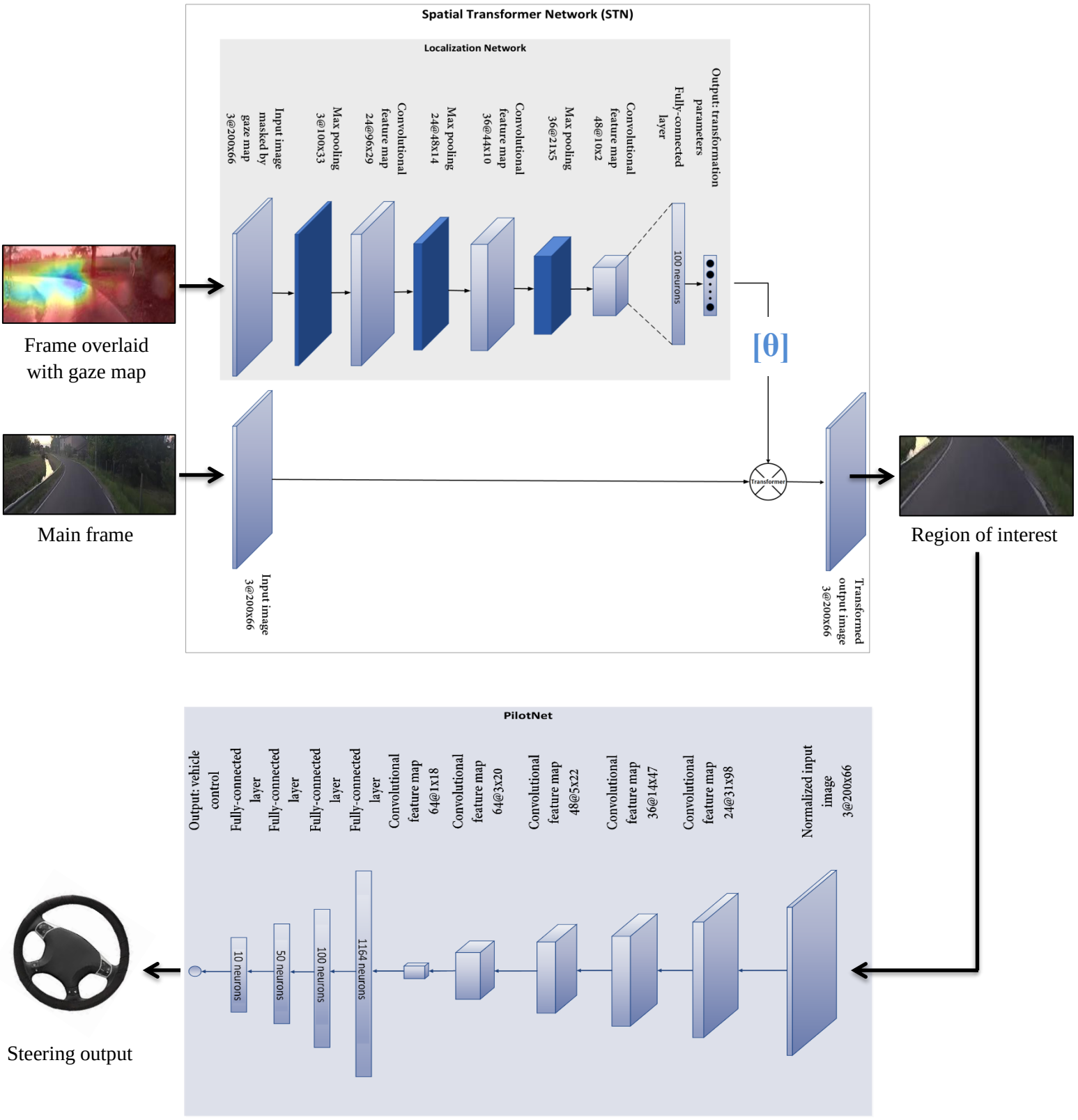

Figure 32 final structure of our STN architecture

6.3.3 Results analysis

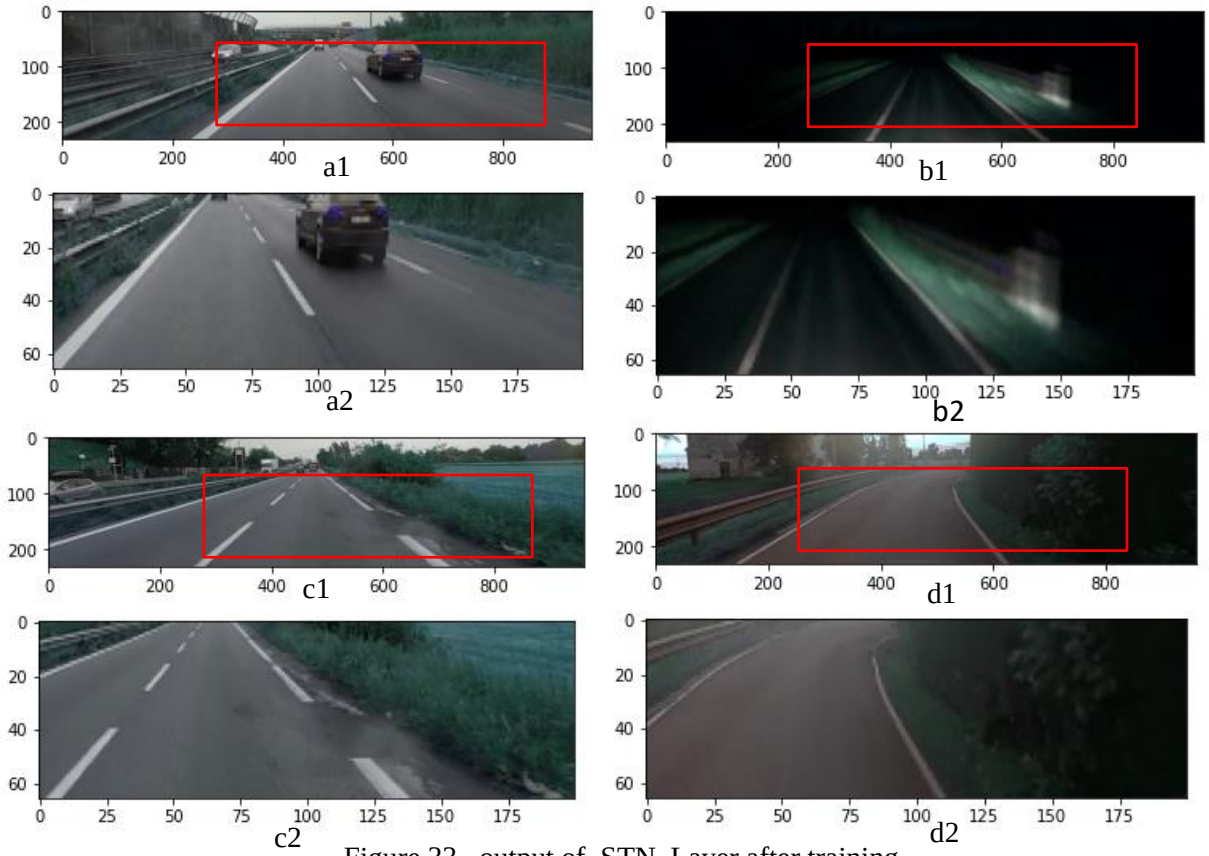

Figure 33 output of STN Layer after training

As shown in Fig.33, the STN Layer learned to focus on the street, and specifically on the lanes. It also tends to shift right a little bit to get the right lane as its more important than the left one and nearer in most cases as the cars always take the right path in 2-ways streets. We can observe that the localization Net of the STN Layer after training outputs nearly the same transformation with small changes from one frame to another, which makes the place and the size of the red box shown in Fig.32 the nearly the same for different input frames. This behavior is because of the position dependence nature of our problem, as a shifted input frame to the right or the left indicated different car positions in the street which leads to different steering angle output.

This architecture scored an RMSE value of **0.008312** which is an improvement of **20.83%,** and to our knowledge, it's the first architecture to use the STN in self-driving cars regression problem, and also the first to use the gaze information.

Its only drawback is its inference time which has a huge bottleneck delay as it gets the gaze map frame from the Gaze Network (discussed in Gaze Network chapter) which is not a real-time network. Its inference time after our modification is approximately 30 sec./frame (Google-Colab GPU). So it will be impossible to be used on embedded systems in real-life SDC's.

6.4 Multi-task Learning (MTL) Architecture

We wanted to make use of the available dataset broadly and let our model generalize better on our original task which is predicting steering angles. We went to the Multi-Task Learning (MTL) approach, which is a successful approach in the field of Natural Language Processing (NLP), drug discovery, speech recognition[0], but most importantly computer vision which is our main use of interest. We can effectively optimize more than one loss function for various tasks, as long as these tasks are somehow correlated, the chance of getting an auxiliary task will help improving the optimization process and hence, our main task performance[26].

In our case we worked on only two tasks: predicting steering angles and predicting the eye-gaze location, and of course, we could add more auxiliary tasks, but this would affect our runtime in training and testing.

In MTL, we followed the Hard parameter sharing [27], which means we extract some features by shared layers between all tasks, imagine it as the backbone of the network, then input the extracted features to the heads of the network which represent task-specific layers, Fig.34, this kind of sharing, reduces the risk of overfitting[28].

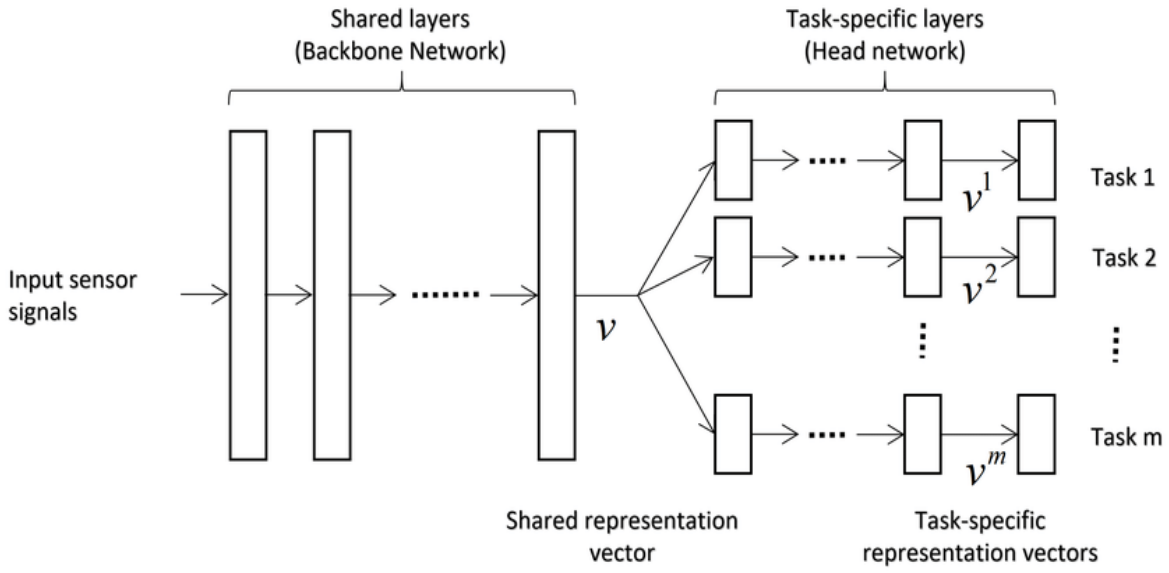

Figure 34 MTL general schematic

Concerning our multitask architecture, we needed two heads one for predicting the steering angle which is the (main task) and the other is for predicting the gaze map (secondary task), and it's supposed that adding the secondary task will improve the primary target task as discussed above.

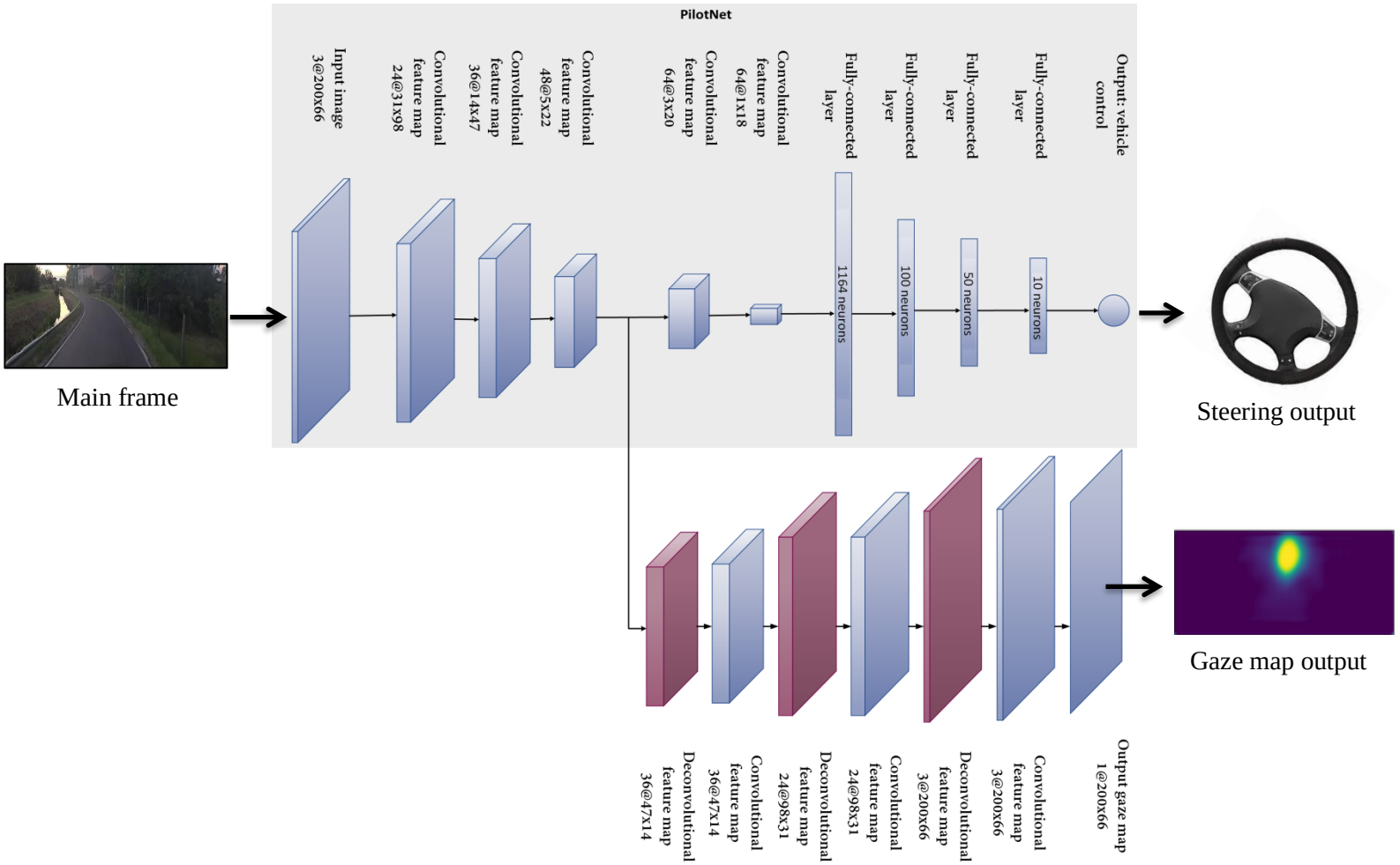

Figure 35 Multi-task architecture

Our Multi-task architecture idea is to use the PilotNet as the main block, as our primary task is the steering angle prediction. We use the first 3 convolutional layers of the PilotNet as the shared layers. Afterward, it is branched into 2 branches, one is the rest of the PilotNet and the other one is a decoder like network consisting of 3 deconvolutional layers inverting the effect of the 3 shared layers to get the same size of the input image for gaze map. these 3 deconvolutional layers have 3 convolutional layers between them to increase the number of image constructing parameters and improve the accuracy of the constructed image as shown in Fig.35.

While each task has its own defined loss function: the steering branch loss function is the root mean squared error between the original and predicted steering angle, while the gaze map branch is pixel-wise RMSE between the ground truth gaze map and the predicted one. We had to define a loss function for the two tasks together, so at first, we combined the different losses simply by summing them into one loss function, but this ended up with bad results as one task dominates the overall sum, because of different loss scales. So we replaced the sum of losses to a weighted sum of losses with steering branch having full weight, and the gaze branch multiplied by 0.1 as our main task is the steering one so giving it higher loss value makes the system focus on this task more than the other, and the results were better.

6.4.1 Results Analysis

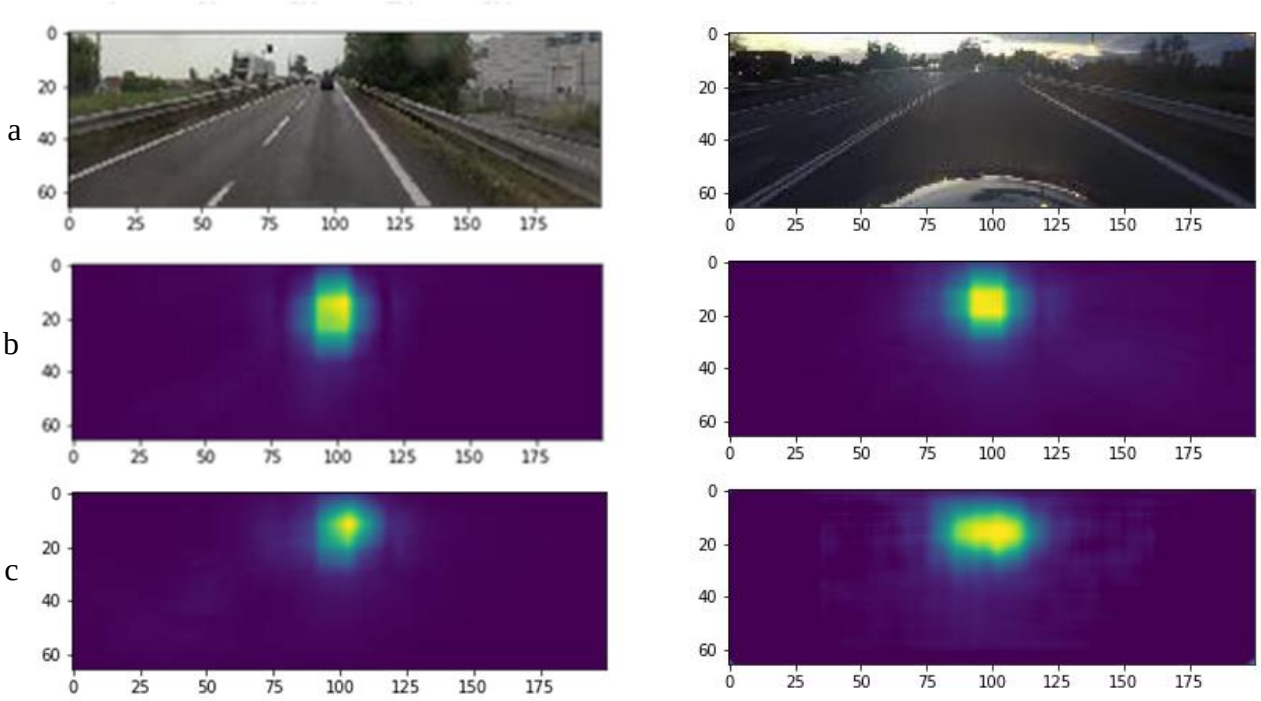

Figure 36 From top : input frame, Gazemap groundtruth, and prediction of gaze branch

In Fig. 36, (a) is the input frame, (b) is the ground truth of gaze map, and (c) is the prediction of the gaze branch. We can see that the network makes very good predictions, and it learned to detect the far end of the street which helped in the prediction of the steering angle better and improved the accuracy.

This architecture scored an RMSE value of **0.008312** which makes an improvement of **21.42%** over the baseline. The main advantage of this MTL architecture is the inference time. The previous architectures have the Gaze network put in serial before the model, which takes the input frame and generates the gaze map which will be used by the network as an input. The inference time of the Gaze network is very large as discussed before, it's equal approximately **24** sec/frame. This MTL architecture replaces the Gaze network completely, and its inference time is **0.0028** sec./frame, which gives us a very high frame rate to be used in real-time embedded system devices.

6.5 STN + Multi-task Learning Architecture

Although our STN attention model is very powerful as it seeks only the important part of the input image and made a very good improvement, it has 2 problems. The input gaze map is not very effective to the network as the network can't make large changes in the transformations as discussed before, and it didn't make the optimal use of the gaze information. The second problem is that the network gets the gaze information as an input which makes the delay time of the network very high because of the gaze network which predicts the gaze map for the input frame. As a solution to these problems and to increase accuracy, we thought of merging the 2 architectures: STN and Multi-task.

In this architecture Fig.37, we combine the strengths of the 2 architectures by letting the network with its 2 branches decide the best regions for the given tasks. We tried three forms, we let the STN layer make 1, 2 and 3 transformations on the input frame to give it more degrees of freedom, then we stacked the output images into a single frame and resize it to be of size 200x60 image to be the same input size to the multi-task network. In the 1-transformation trial, it was hard for the network to focus on only one region in the input frame, as each one of the 2 tasks wants the STN layer to focus on a different part of the frame. The steering branch pushes the STN layer to focus on the right lane as discussed in the attention section, but the gaze branch pushes it to focus on the far end of the road to get the gaze position as it's mostly at the far end of the road. So 1 transformation was not enough. We added 2 transformations which gave the network one more degree of freedom. And a slight improvement was achieved by increasing the transformations from 2 to 3 transformations.

The initial value for the STN layer transformation is better to be such that the whole frame stays the same and not to be changed, and let the network learn the best transformation as recommended by the authors of [25], but here we made a random small change between the 2 or 3 transformations to prevent getting the same transformation and to get better results.

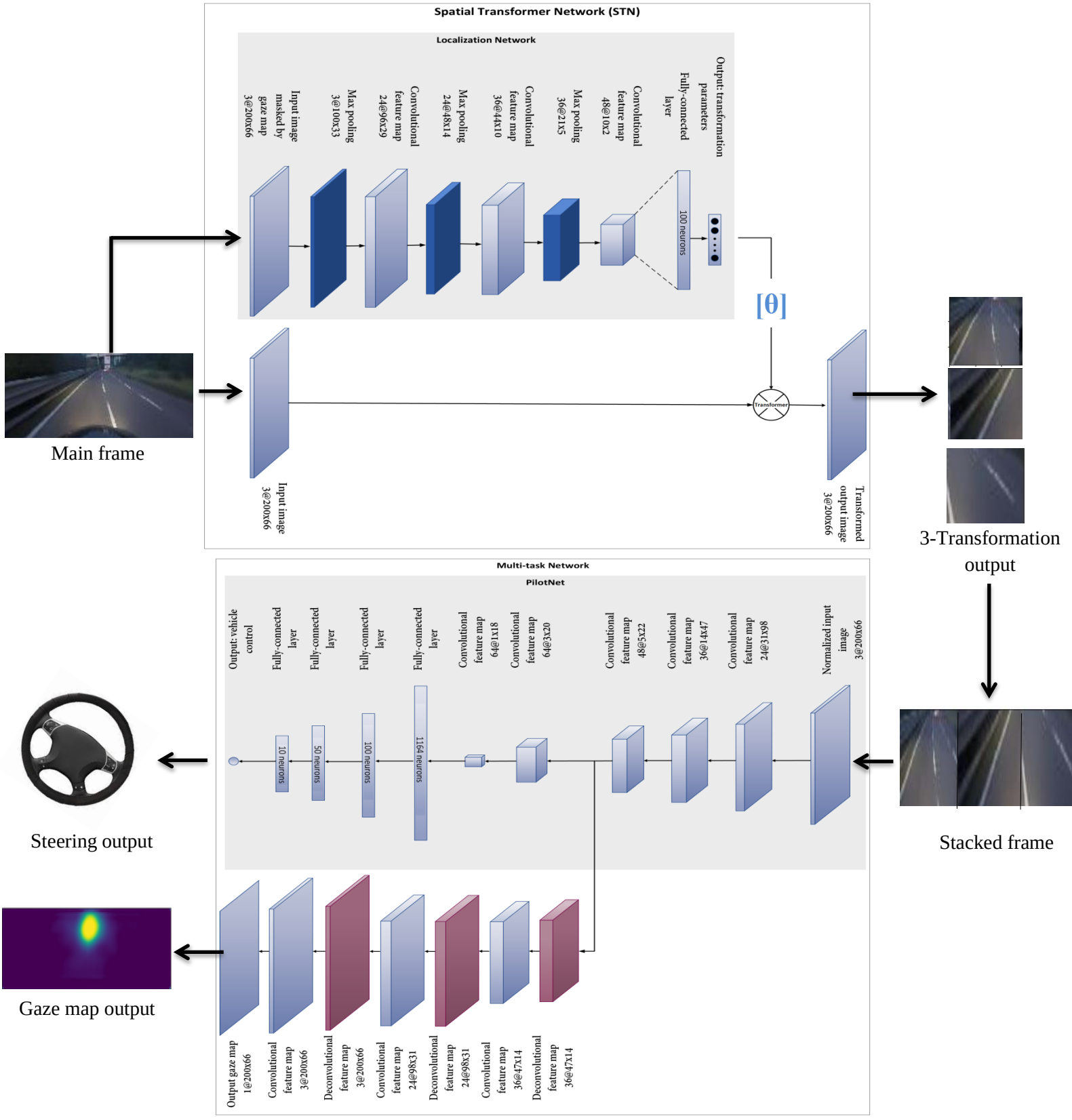

Figure 37 Multi-task + STN Final architecture (3-Transformations)

6.3.1 Results Analysis

In Fig.39, the results of the 2-transformation version are shown, we can notice that each one of the 2 tasks forces one of the 2 transformations to focus on its preferred region of interest, so the network learned to focus on the right lane and the street's far end for the 2 transformations. We can see that it's not always a good region to crop for situations like the one shown in Fig. 20-b-1, but the network learned the best place to focus on from the whole training set, and this can be explained as the human driver from his experience in driving, expects to have the right lane for example at specific place relative to the car's position and he tries to vary the steering angle until he gets the right lane in the place he decides and keep changing the steering angle to have the lane position relative to the car constant as the street curvature changes.

We can see the gaze branch learned to detect all the paths the car can take in Fig. 39-e-1, the ground truth has only one spot tending to the right path as the gaze network has much information more than ours like optical flow and semantic segmentation besides a sequence of 16 frames, but learning the paths is important and sufficient to improve the steering angle accuracy.

The three transformations are shown in Fig. 38, the network learned to focus on the right lane and the street far end like the 2 transformations–Network, in addition to the third one is focused on the left lane which is less important than the right one.

This Novel architecture combining STN and MTL scored an RMSE value of **0.008312** which makes an improvement of **32.37%** over the baseline for the 2-transformations and scored an RMSE value of **0.006701** which makes an improvement of **36.18%** over the baseline for the 3-Transformations which is the best result achieved in all the architectures. It also has an inference time of 0.015 which makes a frame rate of 66 frames/second. The delay time is directly proportional to the number of the transformations used, so there is a tradeoff between inference time and getting less RMSE value. Increasing the number of transformations also decreases the quality of each frame in the final stacked frame. For visualization for our model please refer to our video demo at *https://goo.gl/13551M*

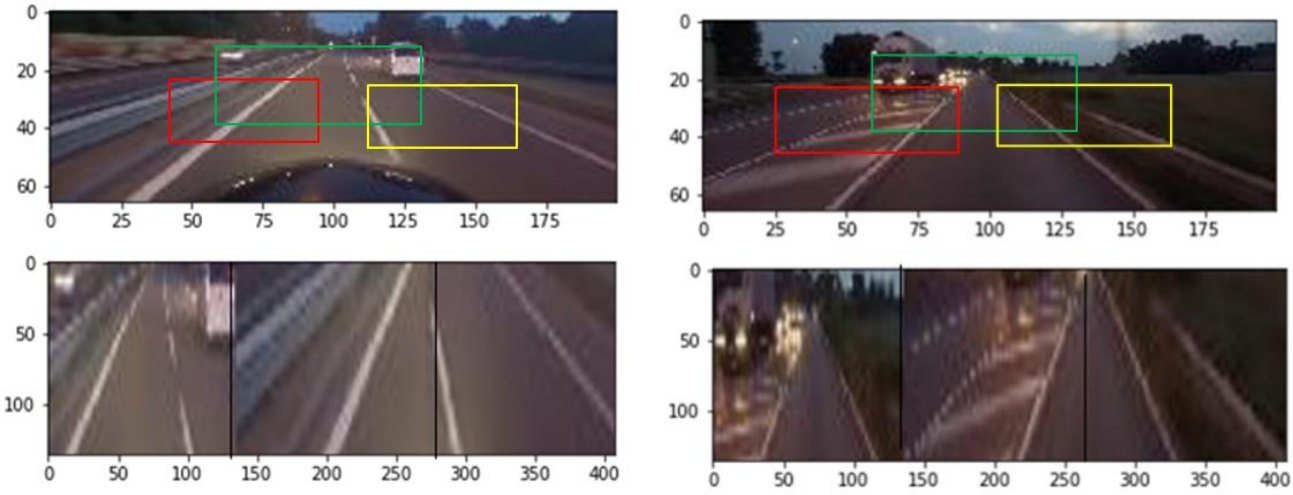

Figure 38 3 transformations STN layer output

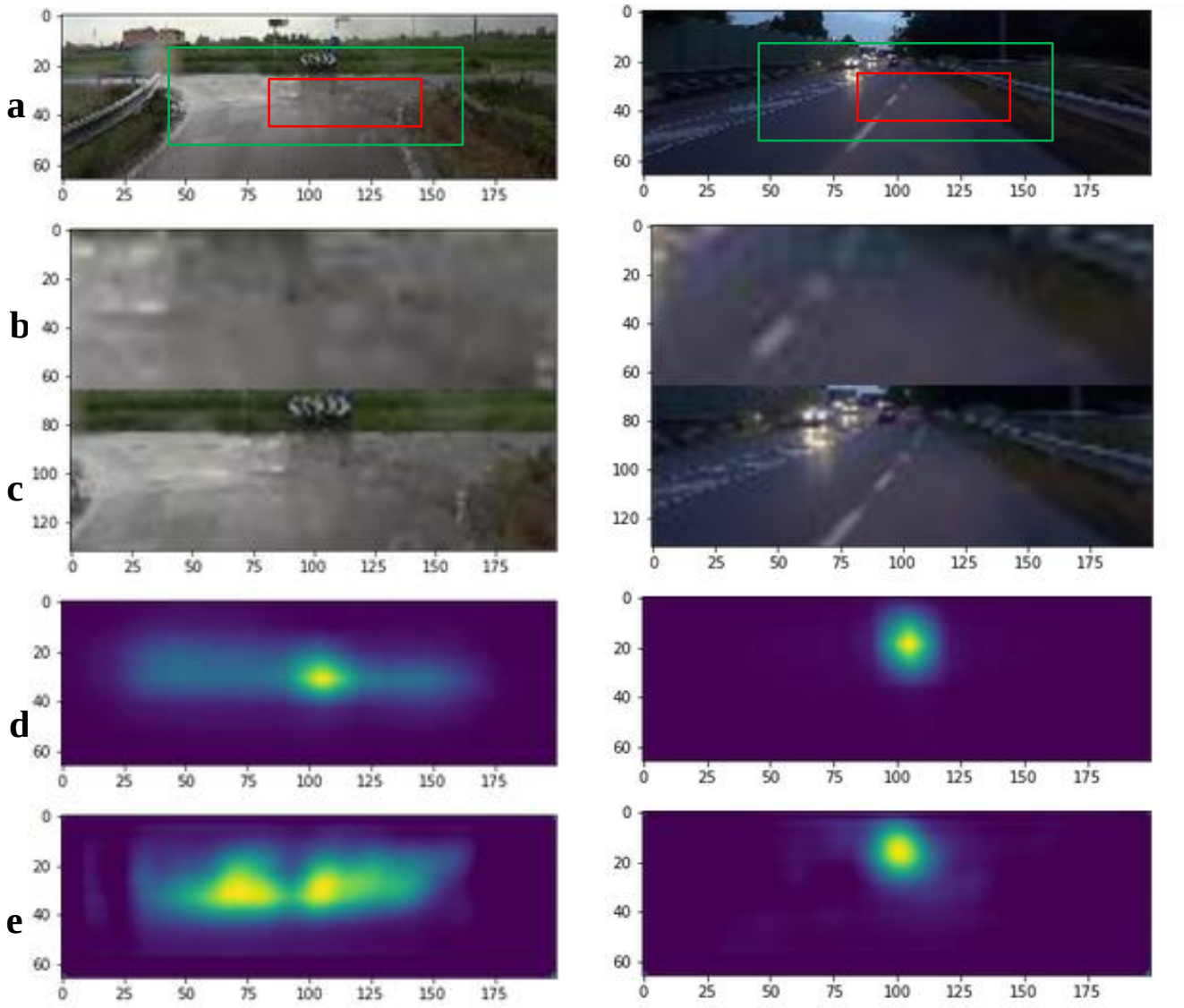

Figure 39 output of STN layer (b/c), gaze ground truth (d) and predictions of gaze branch (e)

6.4 Results

The results of the discussed architecture are summarized in the following table:

Table 4 Results summary

Architecture	RMSE	Improvement (%)	Inference time (sec.)
Baseline	0.0105	-	0.0028
Middle-fusion	0.0071132	30.45%	27 + 0.00858
STN layer	0.008312	20.83%	27 + 0.005
Multi-task	0.008250	21.42%	0.0028
STN+MTL 2-Trans	0.007101	32.37%	0.01
STN+MTL 3-Trans.	0.006701	36.18%	0.015

6.5 On-Road Test

To insure the ability of the model to generalize well in different distribution unseen environment we did a real time road-test in our city in Egypt, all the preprocessing needed were: right & left shifting of new frames to compensate for the different road and lane sizes between the country of the dataset and our country besides a “histogram matching ” to match the color distribution between our frames with a reference frame from the dataset of training, the STN made it possible that the model can really drive in this very different unseen environment quit well without fine-tuning of, as the frame cropping and zooming enables us to shift the image without adding black boxes in the side to compensate this shift to preserve the frame size , and it made the histogram matching possible, because the zoomed area is always focused on the road and the lanes , so there is nearly only 2 main colors and no more objects with different colors besides the road appear in the frame which can vastly change the color components of each frame and make it very hard to predict and compensate for this differences. histogram matching example is shown in the below figure.

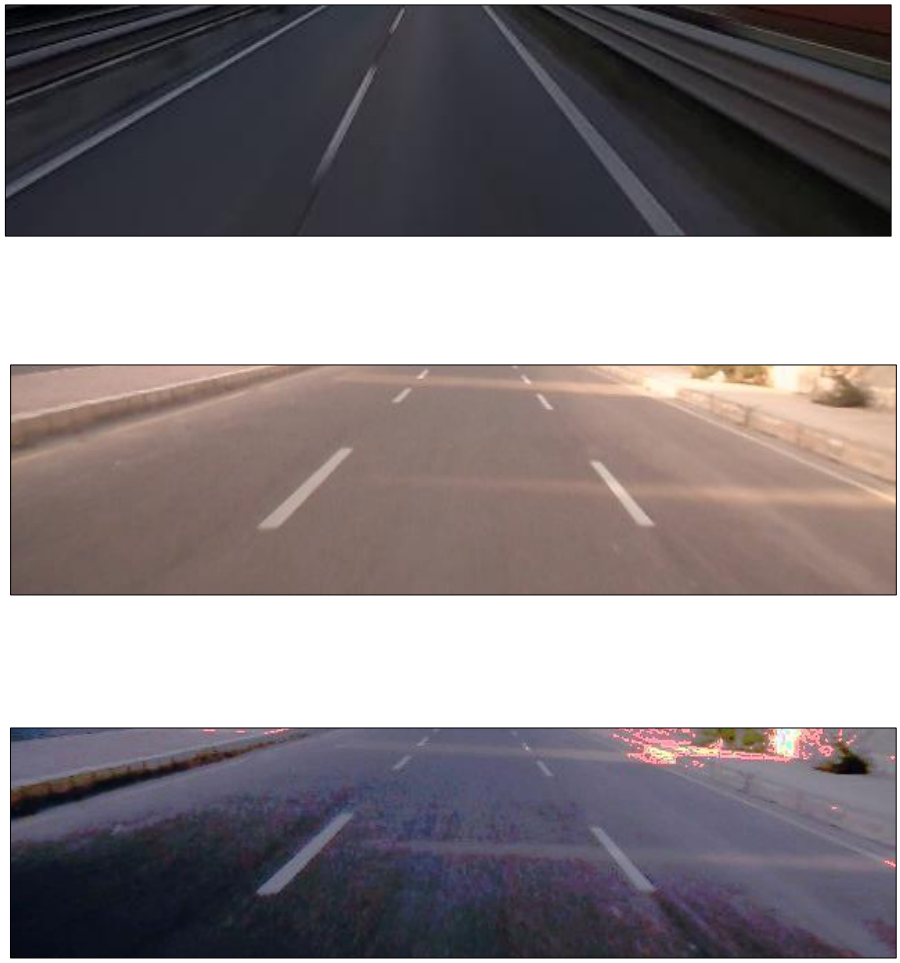

Figure 40 From top: reference frame, new environment frame, new environment frame after applying histogram matching with the reference frame

Chapter 7

Conclusion

We have verified that incorporating eye gaze information into the training process does help the model to predict accurate steering commands, generalize to unseen environments, and operate in different environments without being explicitly trained in them.

We introduced an attention mechanism that enables the model to focus on the important features in the road such as the lanes and eliminate unnecessary elements. We also used multitask learning (MTL) and implement a multitask network which can make steering predictions and generate gaze maps.

We introduced a state of the art architecture that combines attention mechanism and multitask learning to predict steering commands and gaze maps, achieving an improvement percentage of 36% over the Nvidia PilotNet.

Due to the nature of our problem, being able to operate in real-time is a must. We have optimized the inference time to reach 0.015 seconds/frame.

We test our model on a recorded dataset from Egypt and the model was able to predict accurate steering commands without any fine-tuning which is proof of its ability to generalize to unseen environments.

References

[1] Road traffic injuries from WHO, https://www.who.int/news-room/fact- sheets/detail/road-traffic-injuries

[2] Preparing a Nation for Autonomous Vehicles: Opportunities, Barriers, and Policy Recommendations, 2010.

[3] J3016-201401 Standard for Terms Related to On-Road Motor Vehicle Automated Driving Systems.

[4] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst. At Google Brain & Waymo, 2018.

[5] Liu, C., Chen, Y., Tai, L., Ye, H., Liu, M., & Shi, B. E. (2019). A gaze model improves autonomous driving. Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications - ETRA '19.

[6] Chen, Y., Liu, C., Tai, L., Liu, M., & Shi, B. E. (2019). Gaze Training by Modulated Dropout Improves Imitation Learning. 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

[7] Unsupervised Learning: Foundations of Neural Computation.

[8] Schmidhuber, J. (2015). "Deep Learning in Neural Networks: An Overview.

[9] Hubel, DH; Wiesel, TN (October 1959). "Receptive fields of single neurons in the cat's striate cortex". J. Physiol.

[10] LeCun, Yann; Bengio, Yoshua; Hinton, Geoffrey (2015). "Deep learning". Nature.

[11] Weng, J; Ahuja, N; Huang, TS (1993). "Learning recognition and segmentation of 3-D objects from 2-D images". Proc. 4th International Conf. Computer Vision: 121–128.

[12] Schmidhuber, Jürgen (2015). "Deep Learning". Scholarpedia.

[13] Homma, Toshiteru; Les Atlas; Robert Marks II (1988). "An Artificial Neural Network for Spatio Temporal Bipolar Patterns: Application to Phoneme Classification". Advances in Neural Information Processing Systems.

[14] Waibel, Alex (December 1987). Phoneme Recognition Using Time-Delay Neural Networks. Meeting of the Institute of Electrical, Information and Communication Engineers (IEICE). Tokyo, Japan.

[15] Alexander Waibel et al., Phoneme Recognition Using Time-Delay Neural Networks IEEE Transactions on Acoustics, Speech, and Signal Processing, Volume 37, No. 3, pp. 328. - 339 March 1989.

[16] LeCun, Yann; Bengio, Yoshua (1995). "Convolutional networks for images, speech, and time series". In Arbib, Michael A. (ed.). The handbook of brain theory and neural networks (Second ed.). The MIT Press. pp. 276–278.

[17] Bojarski, Mariusz, et al. "End to end learning for self-driving cars." (2016) arXiv preprint arXiv:1604.07316 .

[18] Palazzi, A., Abati, D., Calderara, S., Solera, F., & Cucchiara, R. (2018). Predicting the Driver's Focus of Attention: the DR(eye)VE Project. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1–1.

[19] Alletto, S., Palazzi, A., Solera, F., Calderara, S., & Cucchiara, R. (2016). DR(eye)VE: A Dataset for Attention-Based Tasks with Applications to Autonomous and Assisted Driving. 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW).

[20] Yu, Fisher and Vladlen Koltun. (2016) ."Multi-Scale Context Aggregation by Dilated Convolutions." CoRR

[21] Ferrari, V., Hebert, M., Sminchisescu, C., & Weiss, Y. (Eds.). (2018). "ICNet for Real-Time Semantic Segmentation on High-Resolution Images"Computer Vision – ECCV 2018.

[22] Bojarski, Mariusz, et al. "Explaining how a deep neural network trained with end-to-end learning steers a car."(2017) arXiv preprint arXiv:1704.07911.

[23] Vaswani, Ashish, et al. "Attention is all you need."(2017) Advances in neural information processing systems.

[24] Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention."(2015) International conference on machine learning.

[25] Jaderberg, Max, Karen Simonyan, and Andrew Zisserman. "Spatial transformer networks." (2015) Advances in neural information processing systems.

[26] Caruana, Rich. "Multitask learning." (1997) Machine learning 28.1: 41-75.

[27]Caruana, R. "Multitask learning: A knowledge-based source of inductive bias." (1993) Proceedings of the Tenth International Conference on Machine Learning.

[28] Baxter, Jonathan. "A Bayesian/information theoretic model of learning to learn via multiple task sampling." (1997)Machine learning 28.1 :7-39.