International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 07 | July 2021 www.irjet.net p-ISSN: 2395-0072

Autonomous Driving Vehicle with Multitask Learning

Pranav Shivagaje[1^], Ashok Shivagaje[2]

[1^]Department of Electrical Engineering, College of Engineering, Pune
[2]Associate Professor, College of Agriculture, Dhule

Abstract : Self-driving car capabilities are severely impeded by the lack of two prominent neuro-psychological constructs in extant artificial intelligence – perception and anticipation. This project aims at tackling the problem of anticipation by teaching neural networks to acquire domain knowledge.

Domain knowledge acquisition is the holy grail that machine learning endeavours to acquire. However, current machine learning approaches to self-driving limit the complexity of neural networks to single and simple tasks.

This project demonstrates an improvement in the capabilities of extant driving mechanism using multitask learning - a learning paradigm which utilizes a single neural network with several shared and unshared layers to predict several separate and related, yet statistically independent tasks.

Multitask neural networks facilitate the acquisition of domain knowledge much like the human brain, in that they multiplex related tasks in a single domain to the same neural pathways. This multiplexing not only reduces the number of learnable parameters, but the additional tasks make the learning models robust to overfitting.

The project also endeavours to implement the multitask learning paradigm in hardware. A scaled-down electric car, powered by DC motors, was mounted with an on-board raspberry-Pi camera module. During the data collection phase, the pi module was IoT enabled facilitating the collection of training data. During the autonomous telemetry phase, this module exchanged images and corresponding driving commands using a two-way TCP connection with a remote server.

Keywords: CNN, BLDC, Multi-task learning, Gradient Descent, Learning rate, Back Propagation.

Introduction:

Human safety is most important in an autonomous driving system. Existing models that have been deployed are built focused on predicting a single atomic task, with no regard towards other related tasks contributing to the action of driving on public roads. Humans, however, start learning a new task armed with all the knowledge of the things they've already learnt and often learn multiple related tasks together, allowing sharing of neural pathway among tasks. We propose to do the same by building a NN with shared representations.

The project is aimed at building a robust, end-to-end autonomous driving system. The proposed model improves prediction accuracy over existing models by using Multitask Learning to acquire domain knowledge in the field of driving. This method is chosen so that a model that is impervious to overfitting is developed, that facilitates sharing of common features across the closely related tasks of driving speed and steering angle.

A scaled-down IoT enabled version of a car is built for testing and demonstration purposes. The car is equipped with a Raspberry pi as an on-board controller and pi camera for image capture for autonomous navigation.

The list of objectives is:

1. Identification of related tasks while driving

2. Design of Multitask Network

3. Design of scaled-down car by surveying motors and drivers

4. Building IoT-enabled car using Raspberry pi, pi camera and setting up a wireless connection

5. Data collection and labelling in training environment

6. Tuning hyper-parameters of Multitask Network for collected data set

7. Testing autonomous driving system

Software

1. Jupyter Notebook for writing code in Python and Keras

2. Udacity's open source Autonomous Driving Simulator

3. Proteus for hardware design simulation of car

Hardware

1. Raspberry pi as on-board car controller

2. pi camera that captures images in the car view

3. DC motors for rotating car wheels according to controller commands

4. Motor driver for operating the motors

5. scaled-down chassis of car for housing the hardware

Autonomous Vehicle Hardware

Sensors: Sensors are the components that allow the autonomous vehicle to take in raw information about the environment. The main sensors include GPS/Inertial Measurement Unit (IMUs), camera, LiDar, and radar. LiDar, for example, is great at capturing information in various types of ambient light (whether night or day). Most autonomous vehicles combine the readings

of multiple sensor types to add extra redundancy and compensate for the weaknesses of the various sensors in a process called sensor fusion. Actuators: Actuators are the components of a machine responsible for controlling and moving the system that involve the on-board computer, motors etc.

Autonomous Vehicle Software

Autonomous vehicle software can be categorized into three systems: perception, planning, and control.

Perception: The perception system refers to the ability of the autonomous vehicle to understand what the raw information coming in through the sensors means. It enables the car to understand from a given picture frame whether a certain object is another car, a pedestrian, or something else entirely.

Planning: The planning system refers to the ability of the autonomous vehicle to make certain decisions to achieve some higher order goals. This is how the autonomous vehicle knows what to do in a situation – whether to stop, go, slow down, etc. The planning system works by combining the processed information about the environment with established policies and knowledge about how to navigate in the environment (eg. do not run over pedestrians, slow down when approaching a stop sign, etc.) so that the car can determine what action to take (e.g. overtake another car, how to reach the destination, etc.).

Control: The control system pertains to the process of converting the intentions and goals derived from the planning system into actions. Here the control system tells the hardware (the actuators) the necessary inputs that will lead to the desired motions. For example, an autonomous vehicle, knowing that it should slow down when approaching a red light, translates this knowledge into the action of applying the brakes.

MATERIALS AND METHODS:

Data Generation

Nvidias Self-Driving Car data collection process is a standard one. It involves 3 cameras attached to a car being manually driven on public roads by a human driver. One camera was placed in the center and the other 2 were placed on each side of the car. The steering wheel angle was recorded for each instant. Similarly, the following were captured: Images from Center, Left and Right Cameras, Steering Angle, Speed of the car, Throttle.

Training the Autonomous System

The collected data set can now be used to train an ML model that mimics how the human was driving the car- essentially cloning the drivers behaviour to different road scenarios. This is called Behavioural Cloning. Behavioural cloning is a method by which human sub-cognitive skills are captured and reproduced in a computer program.

A CNN typically has three layers: a convolutional layer, pooling layer, and fully connected layer. The convolutional layer performs a dot product between two matrices, where one matrix is the set of learnable parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive.

Gradient descent is an optimization algorithm used to find the values of parameters (coefficients) of a function (f) that minimizes a cost function (cost).

Nvidia's network consists of 9 layers- a normalization layer, 5 convolutional layers and 3 fully connected layers. The first layer normalises the image. The normalisation values are hardcoded and it is not trainable. The convolutional layers perform feature extraction. The 1st three layers use strided convolutions with a 55 kernel, and the last two convolutional layer use non-strided convolution with a 33 kernel size. The convolutional layers are followed by three fully connected layers. The fully connected layers are supposed to act as a controller for the Autonomous system.

Multitask Learning

Most extant learning models exploit statistical relationships between the variables in the data with the sole focus being on a single output derived from the input. The scope of such models is narrow in that they are not reusable, are inflexible and are prone to significant variance error. Human cognitive systems, however, use knowledge acquired while learning one task to learn other different, yet related tasks. Multitask Learning aims to mimic this by having a single neural network predict more than one outcome by using nets which perform separate, yet related tasks by sharing representations. As a result, what is learned for each task helps the other tasks be learned better. A multitask network improves the ability of the NN to generalize across tasks by sharing domain knowledge among related tasks through inductive learning. Here, the training signals of related tasks serve as an inductive bias.

NEURAL NETWORK DESIGN:

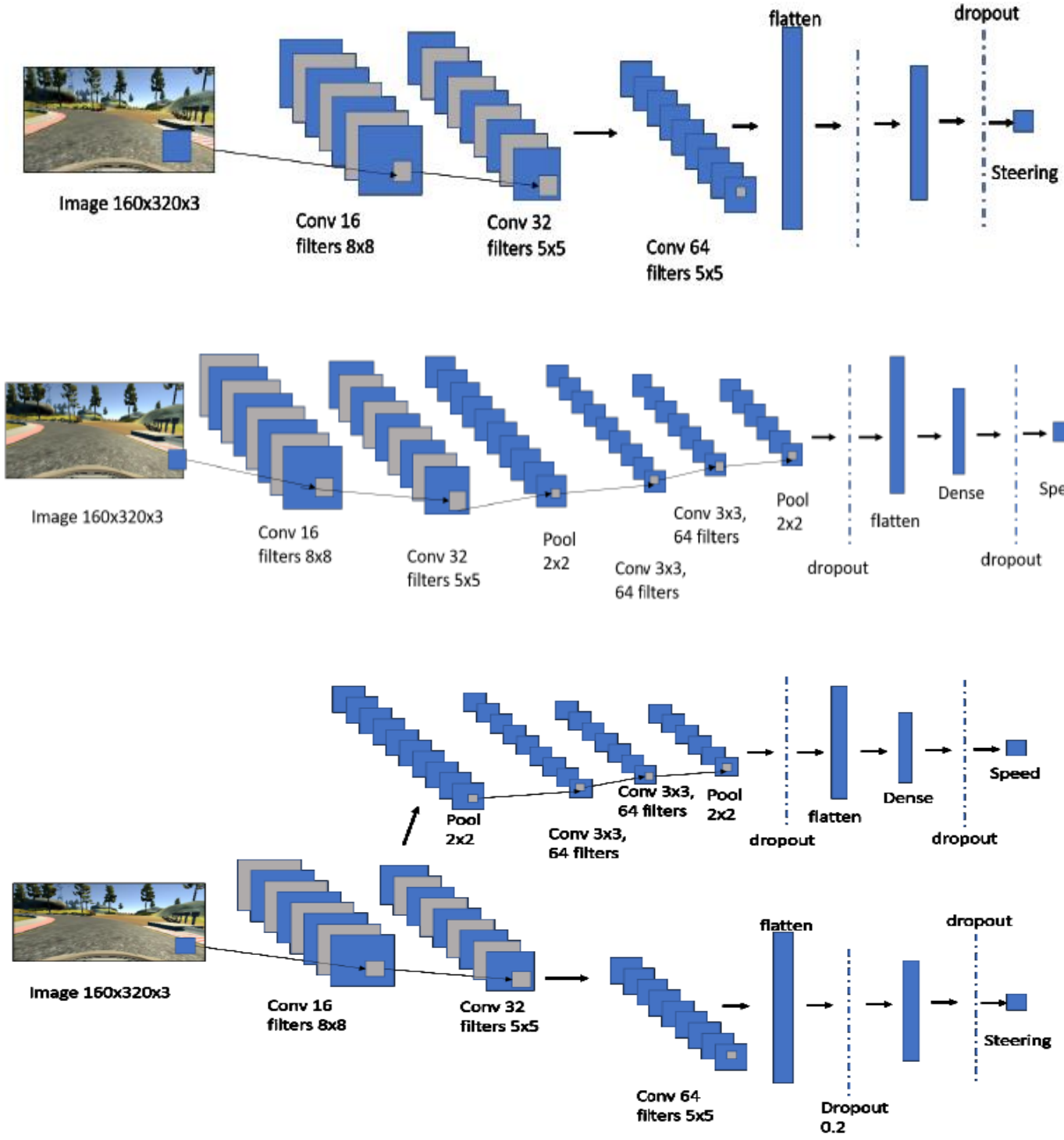
Ordinary convolutional neural networks have been spearheading self-driving efforts in the AI portion of the technology stack. Convolutional neural networks extract useful features using filters and use these representations to perform specific tasks like steering, speed control, acceleration. The additional learnable filters not only reduce computational complexity but reduce the time required during training. The weights of the filters are also learnable, making feature selection an automated data-driven task.

Multitask learning paradigms are roughly modelled on the human brain's ability to multiplex several tasks to the same neural pathways. MTL involves using a single neural pathway to perform several separate, related yet computationally uncorrelated tasks.

Our MTL architecture involved combining the neural pathways reserved for steering angle prediction and speed prediction into a single neural network with shared and task dependent layers. In Fig 5.3, the first 2 convolutional layers are used to obtain features common to both steering and speed prediction tasks. The succeeding layers are task dependent. The final output for each of the task was a single scalar. The neural network was trained on the mean squared error loss metric. The loss for both tasks was weighed equally and the joint loss function was as follows.

Loss = A[yste^ering ysteering]2 + B[ysp^eed yspeed]2 + jW2

ij j



Collecting training data in Simulated environment. For software testing of the designed Multitask Network, Udacity's open-source Self Driving car simulator was used. The simulator uses Unity (across-platform real-time video gaming engine developed by Unity Technologies) to render graphics to create an artificial environment wherein it allows the user to drive a virtual car on a winding road. The user can control the car via a keyboard or joystick. The car has three virtual cameras mounted on the front, left and right side of the dashboard. Thus, three images (Centre, Right, Left) were captured each time at the rate of 10fps while training. These images, with their timestamp and the corresponding driving commands from the user were used to create a driving log for training the NN. While collecting the data set, the car was driven 10 rounds of the track.

The simulator runs in two modes-

Training mode: Here, it records the camera images and user's driving commands to create a driving log

Autonomous mode: Here, it uses a Python ML program (that the user builds) to show how the car would navigate using the predicted driving commands.

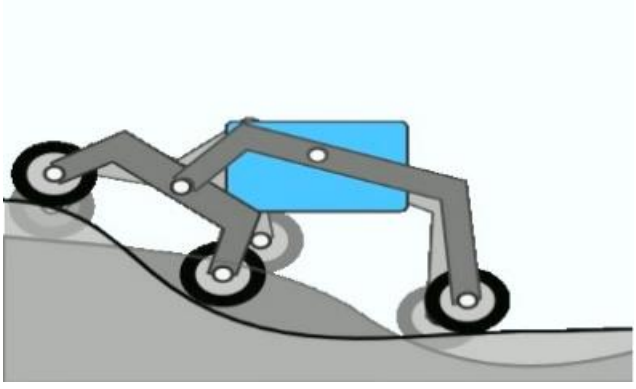
Two tracks are already available on the simulator. More can be constructed via Unity. The simulator follows a client-server architecture where it acts as a server for a client Python program that users write. The program can connect to and receive a stream of image frames from the simulator. The program then uses a machine learning model to process the road images to predict the best driving instructions, and sends them back to the server. Each driving instruction contains a steering angle and an acceleration throttle, which changes the cars direction and the speed (via acceleration). As this happens, the client program receives new image frames at real time. A Multitask Network model was developed as Client program for the Udacity server.

Hardware Design

This mechanism is used to overcome rough terrains while maintaining stability. It was chosen so that the camera and controller mounted on the prototype function unharmed.

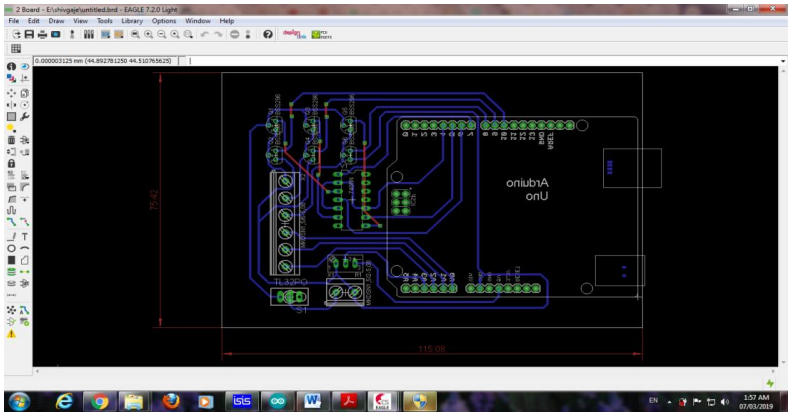
The term "rocker" refers to the larger link on each side of the suspension system. The rockers are connected to each other and the vehicle chassis through a shaft. Relative to the chassis, when one rocker goes up, the other goes down therefore the chassis maintains average pitch reducing vibration and jerks, thereby reducing wear and tear of onboard equipment.

PVC pipes were cut to size and a chassis was constructed using these. PVC pipes were selected as they are lightweight, yet robust.



BLDC Motor Specifications Model: A2212/10T/13T Motor KV: 1000 Maximum Efficiency: 0.8 Max. Efficiency Current: 4-10A No Load Current: 0.5A @10V Current Capacity: 12A/60S LiPO Batteries: 2S-3S By controlling the applied voltage, the DC motor can be run at the base and below the rated speed, whereas by flux weakening above it can be run above the rated speed. Control unit design can be implemented with the use of several analog controllers to digital.

In many of the cases, digital-integrated circuits with Pulse Width Modulation technique (PWM) are implemented to have control of this motor. This speed control can be open or closed-loop control.



In an open-loop control, the input voltage applied is controlled to get the variable speed using different type of controllers like current limiters, potentiometers, chopper circuits, etc. In closed-loop control, the supply voltage is controlled by taking actual speed of the motor as a feedback signal.

The closed-loop control consists of three basic elements: PWM circuit, sensing circuit and motor driver. PWM circuit can be implemented with timer or programmable controllers like micro controller. Actual speed sensing is possible with different sensors like hall-e_ect sensors, optical encoders, IR sensors, etc. ,and motor driver drives the BLDC motor based on signals from the controller. In an open-loop control, the input voltage applied is controlled to get the variable speed using different type of controllers like current limiters, potentiometers, chopper circuits, etc. In closed-loop control, the supply voltage is controlled by taking actual speed of the motor as a feedback signal.

CONCLUSIONS:

A Multitask Network with two outputs was designed by modifying the traditional neural network architecture. Two related tasks in the field of driving, namely vehicle speed and steering angle were chosen as outputs of this network. Basic concepts of Computer Vision and Multitask Learning were understood. The proposed MTL net was first tested on the Udacity Self-Driving

Car Simulator with success. Then, it was implemented on a scaled-down, wirelessly connected car powered by DC motors. The car was mounted with a Raspberry pi and pi camera that captured images at 2 fps. In the training phase, images with training labels were fed to the model and gradient descent was applied to minimise error. The r2-score of the training phase was was 0.93. In the testing phase, the captured images were sent to a computer with the trained MTL model over a two-way TCP connection, getting driving commands by the model. The r2-score of the testing phase was 0.92. The proposed model was shown to perform better than traditional Single Task Networks.

FUTURE WORK:

The proposed driving system predicts the speed and steering angle of the car based only on the current image as seen by the camera. Human drivers, however, anticipate what they might see next in front of the car by analysing a sequence of images that lead up to the current image. This ability to expect a certain input by using past inputs allows humans to prepare for their next action. This ability to anticipate can be incorporated in the proposed MTL network through the use of Long Short Term Memory (LSTM) units, that are structured to take a series of sequential images and then predict the outcomes based on them.

The proposed system was built by taking only a single camera feed directed in front of the car. Two other cameras can be used (one on the left of the car and the other on the right) to provide a more enhanced view of the driving environment. Additional sensors such as ultrasonic sensors for estimating obstacle distance, LIDAR sensors and radar systems can be incorporated to increase the robustness of the system.

Auxiliary tasks such as detection of road signs, pedestrian identication etc. can be added to the MTL network to boost the performance of the two main tasks by acting as additional regularization. This may improve performance and reduce

training time.

REFERENCES

[1] Caruana, R. Machine Learning (1997) 28: 41. https://doi.org/10.1023/A:1007379606734

[2] Shai Ben-david , Reba Schuller. Exploiting Task Relatedness for Multiple Task Learning (2003)

[3] Argyriou, A., Evgeniou, T. Pontil, M. Mach Learn (2008) 73: 243.

https://doi.org/10.1007/s10994-007-5040-8

[4] Guillaume Obozinski , Ben Taskar. Multi-task feature selection (2006).

[5] Sebastian Ruder. An Overview of Multi-Task Learning in Deep Neural Networks

[6] Z. Yang, Y. Zhang, J. Yu, J. Cai, J. Luo, "End-to-end multi-modal multi- task vehicle control for self-driving cars with visual perception" in CoRR, 2018.

[7] Liebel L, Korner K, Auxiliary Tasks in Multi-task Learning, 2018.