Object Detection For Simulated Autonomous Cars Using Modified LeNet Model

Madhur Chhangani[#1], Pradyot Patil[#2], Prasad Pattiwar[#3], Priyanka Singh [#4], Purvesh Baghele [#5], Sai Pradyumn Shrivastava[#6] and Shailendra Aote[#7]

[#] *Department of Computer Science and Engineering, Shri Ramdeobaba College of Engineering and Management, Nagpur, Maharashtra, India.*

[1]chhanganima@rknec.edu
[2]patilps@rknec.edu
[3]pattiwarpv@rknec.edu
[4]singhps4@rknec.edu
[5]baghelepk@rknec.edu
[6]shrivastavas1@rknec.edu
[7]aotess1@rknec.edu

***Abstract* – Artificial Intelligence has evolved through the years to have applications in new domains such as object detection for autonomous cars. The accuracy required by these domains often exceeds the accuracy provided by traditional neural net models. An emerging alternative for training self-driving cars can be provided by a modified LeNet model. The aim of this paper is to provide a modified LeNet model to further increase the accuracy. The succeeding topics explore the proposed solution to build, train, and test our modified LeNet model and use it to simulate self-driving car and incorporate object detection to get a better simulation of real time traffic scenarios.**

***Keywords* – *Autonomous cars, object detection, car simulator, LeNet model, behavioral cloning*.**

I. BACKGROUND

An Autonomous Car [1], also known as a self-driving, robot car, or driverless car, is a vehicle that is capable of sensing its environment and moving with little or no human input. Autonomous cars combine a variety of sensors to perceive their surroundings, such as radar, Lidar, sonar, GPS, odometry and inertial measurement units. Sensory information is given as input to complex models to identify appropriate navigation paths, as well as obstacles and relevant signage. One way to identify appropriate navigation paths, obstacles and relevant signage is to use object detection.

Object detection [2] is the process of finding instances of real-world objects such as faces, bicycles, and buildings in images or videos. Object detection algorithms typically use extracted features and learning algorithms to recognize instances of an object category.

Convolutional Neural Networks [3] are special kind of multi-layer neural networks. Like almost every other neural networks they are trained with a version of the back-propagation algorithm. Where they differ is in the architecture. Convolutional Neural Networks are designed to recognize visual patterns directly from pixel images with minimal pre-processing. They can recognize patterns with extreme variability (such as handwritten characters), and with robustness to distortions and simple geometric transformations.

Convolutional neural networks have been associated with self-driving cars largely. In end-to-end learning system for self-driving cars, weights of the network are trained to minimise the mean-squared error between the steering command output by the network and the command of either the human driver or the adjusted steering command for off-centre and rotated images.

Behavioral cloning is a method by which human sub-cognitive skills can be captured and reproduced in a computer program. As the human subject performs the skill, his or her actions are recorded along with the situation that gave rise to the action. A log of these records is used as input to a learning program. The learning program outputs a set of rules that reproduce the skilled behavior. This method can be used to construct automatic control systems for complex tasks for which classical control theory is inadequate. It can also be used for training. Humans often learn how to perform tasks via imitation: they observe others perform a task, and then very quickly infer the appropriate actions to take based on their observations. Behavioral cloning is an attempt to observe and imitate.

II. INTRODUCTION

This part of the paper discusses about basic concepts of technology related to autonomous cars and the proposed model-based solution to object detection.

A. LeNet Model

The LeNet architecture was first introduced by LeCun et al. in their 1998 paper, “Gradient-Based Learning Applied to Document Recognition” [4]. As the name of the

paper suggests, the authors' implementation of LeNet was used primarily for OCR and character recognition in documents.

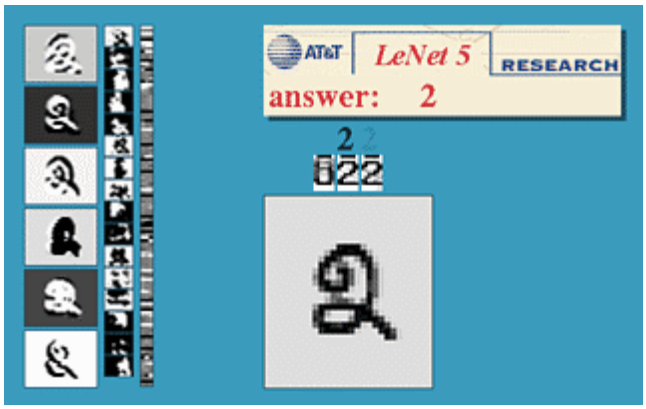

Figure 1. LeNet Model training example

The LeNet architecture is straightforward and small neural network architecture. The LeNet model is also known as LeNert-5. LeNet-5 was designed for handwritten and machine printed character recognition. In this paper, we further explain our modified LeNet model in the implementation section.

B. Udacity's Self-Driving Car Simulator

We have used Udacity's [5] self-driving car simulator for the purpose of simulating autonomous car and training the convolutional neural network. This simulator was built for Udacity's Self-Driving Car Nanodegree, to teach students how to train cars how to navigate road courses using deep learning. The simulator contains two tracks. The simulator helps us in training our model on one track and then using behavioral cloning to simulate autonomous driving on the second track. The environment simulator is powered by Nvidia GPU and Unity. The Simulator has two modes: Training Mode and Autonomous Mode. In the training mode, we drive the car manually to record the driving behavior. We then use the recorded images to train our learning model. In the autonomous mode, we test our neural network learning model to see how well our model can drive the car without dropping off the road. The simulator is acts as a server where our program can connect to and receive a stream of image frames from it.

III. RELATED WORK

Nvidia in its famous paper – "End to End learning for self-driving cars" [6] devised a process of recording training images from three sides of the car viz. left, centre, and right. Convolutional neural networks were incorporated for training. CNNs are used widely for image recognition because the convolution operation captures the two dimensional nature of images. By using convolutional kernels to scan images, relatively few parameters need to be learned compared to total number of operations.

IV. IMPLEMENTATION

This part of the paper describes our approach for simulating a self-driving car that is trained using a neural network which uses a modified version of LeNet model as well as using a similar simulation to detect objects in real time traffic.

1. Training the neural network

We collected training images from Udacity Self-driving car simulator (figure 2). Along with the images, additional data such as speed, throttle and steering angle was also collected. The data set was split into 3511 training samples and 878 validation samples. This data set was clearly not sufficient to train the model and gain a good accuracy while testing hence we used image augmentation techniques, viz. zooming, panning, flipping, brightness alteration to expand the data set (fig 3). This was also done in order to break the continuity of images of the track and add a sense of randomness to the data so that the model actually trains itself rather than memorize the data. The idea is to predict the steering angle for a given image.

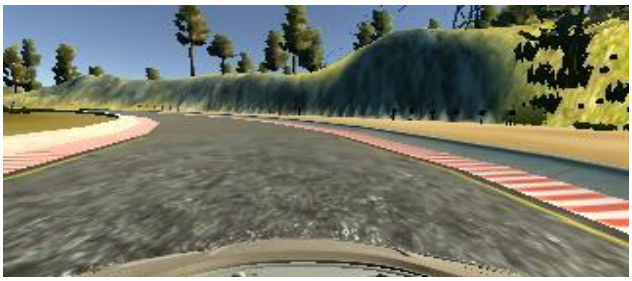

Figure 2. Training image

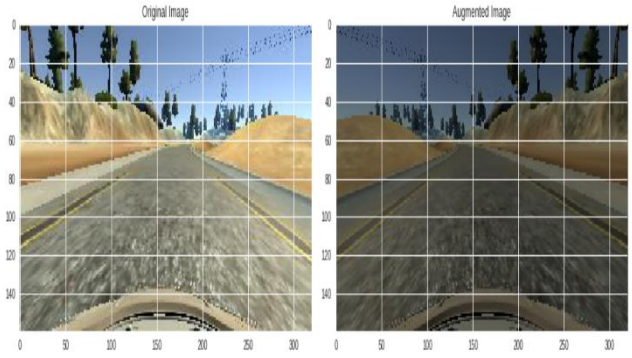

Figure 3. Original image (left) vs Flipped and brightness altered Augmented image (right)

1.1 Modified LeNet model

The first layer of our sequential model is a Convolution2D layer which has 24 filters of size 5 x 5 with a stride of kernel as 2. The input shape of the data is 66 x 200 x 3.

The second layer of our sequential model is a Convolution2D layer which has 36 filters of size 5 x 5 with a stride of kernel as 2.

The third layer of our sequential model is a Convolution2D layer which has 48 filters of size 5 x 5 with a stride of kernel as 2.

The fourth layer of our sequential model is a Convolution2D layer which has 64 filters of size 3 x 3 with a stride of kernel as 1. The fifth layer is same as the fourth layer.

This is then converted into a one dimensional array by adding a Flatten layer so that it can be fed to our fully connected layers that follow. The output shape of the fifth layer comes out to be 1 x 18 x 64.

This is then followed by 3 Dense layers. These dense layers have 100,50,10 input noes respectively. And finally, we add a Dense layer with single output node which will output the steering angle of our car. All these layers have elu [7] as the activation function. This function was chosen over sigmoid function to avoid vanishing gradient problem [8] and over relu [9] because elu function, unlike relu, has a non-zero function in the negative region, i.e elu function always has a chance to recover and fix its weight parameters to decrease its error, meaning it always remains capable of learning and contributing to the model unlike relu which can eventually die.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 31, 98, 24)	1824
conv2d_2 (Conv2D)	(None, 14, 47, 36)	21636
conv2d_3 (Conv2D)	(None, 5, 22, 48)	43248
conv2d_4 (Conv2D)	(None, 3, 20, 64)	27712
conv2d_5 (Conv2D)	(None, 1, 18, 64)	36928
flatten_1 (Flatten)	(None, 1152)	0
dense_1 (Dense)	(None, 100)	115300
dense_2 (Dense)	(None, 50)	5050
dense_3 (Dense)	(None, 10)	510
dense_4 (Dense)	(None, 1)	11

Total params: 252,219
Trainable params: 252,219
Non-trainable params: 0

Figure 4. History of the modified LeNet model.

Optimizer used was Adam optimizer with a learning rate of 0.0001. Furthermore, our model was trained with 10 epochs with each epoch having 300 steps.

2. Object detection

We used tensorflow's object detection API to identify entities in real life traffic such as cars, trucks, pedestrians and traffic lights. Our implementation sends a warning message to the car to stop immediately as soon as an object detection window occupies 60% of the whole window.

V. RESULTS

Training Phase - Udacity Car simulator's training mode was used to capture the training data which contained training images consisting of the views from the point of view of car viz. left, right and center along with steering angle, speed and throttle associated with each image.

Testing Phase - Udacity Car simulator's Autonomous mode was used to test the previously trained driverless car on different terrains by providing the steering angle and keeping the speed constant. To test the trained car with object detection capabilities in real time scenarios we used GTA V, an open world game. The car was able to detect objects such as human beings, vehicles, traffic lights, and sign boards. It is evident from figure 5 that our model doesn't overfit. Final training loss was 0.0401 whereas validation loss was 0.0353.

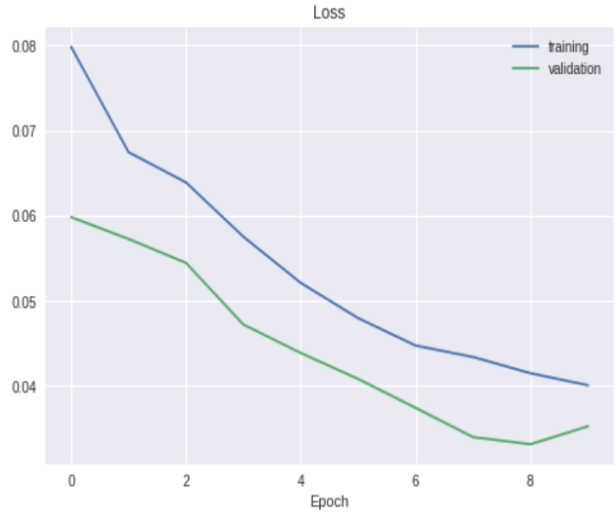

Figure 5. Training loss vs Validation loss.

Figure 6.1

Figure 6.2

Figure 6.3

V. CONCLUSION

Our implementation of LeNet model was able to achieve high accuracy than the traditional approaches. We provided a solution to object detection for autonomous cars by purely using different software techniques involving deep learning, neural networks and machine learning.

VI. REFERENCES

[1] Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Self-driving_car.

[2] MathWorks. [Online]. Available: https://www.mathworks.com/discovery/object-detection.html.

[3] Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Convolutional_neural_network

[4] Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[5] Udacity. (2019, January 11). Udacity/self-driving-car-sim. Retrieved from https://github.com/udacity/self-driving-car-sim.

[6] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang et al., End to end learning for self-driving cars, 2016.

[7] Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Activation_function.

[8] Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems.

[9] Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Rectifier_(neural_networks).