Steering Angle Prediction using Convolutional Neural Networks

Ayush C. Soni
(as2425@g.rit.edu)

***Abstract**: Real-time steering angle prediction is an important task in development of efficient Autonomous Driving systems today. The use of Convolutional Neural Networks by researchers for this task has significantly increased since NVIDIA first proposed an End-to-End CNN architecture called PilotNet to predict steering angles directly from input camera images. Since then, new CNN architectures have been introduced for feature extraction for various tasks in computer vision. In this paper, the use of some of these newer ideas in solving our regression task of steering angle prediction is explored. Specifically, techniques proposed in Darknet-53 architecture of YOLOv3, and Squeeze-and-Excitation networks are used to create CNNs that have similar number of parameters as that of the PilotNet architecture. The performance of these architectures on the Udacity's Self-Driving Car Dataset is then compared with PilotNet model trained on the same data set.*

I. Introduction

Autonomous Driving softwares have evolved more in last decade than ever before with the introduction of deep learning algorithms. One problem are of these systems where deep learning is making a big impact is the problem of accurately predicting steering angles with minimum latency. NVIDIA proposed the use of CNNs in [1], to create an end-to-end system capable of predicting steering angles from raw images of roads taken from a camera mounted on the front hood of a car in real-time. A benefit of such an end-to-end approach is that it doesn't require decomposing the problem into smaller problems such as lane marking and curvature detection and mapping of detected features to the steering control. However, the performance and accuracy of newer CNN feature extractors for various Computer Vision tasks has significantly increased since then, but the usefulness of some of these latest advancements on steering angle prediction hasn't been explored to an extent that's needed.

In this project, two new CNN architectures are created and their performance on the Udacity data set[7] is evaluated. The first is inspired from Darknet-53, a feature extractor used in YOLOv3 paper[2], which achieves better accuracy than other previous architectures that use residual connections such as ResNet and ResNExt. The second is based on the SE RestNet-154 as in [3], that uses the Squeeze-and- Excitation blocks which, in spite of introducing a very small amount of extra computation, give a substantial increase in accuracy. The models created using these two architectures are analyzed and compared with the NVIDIA model, keeping the number of parameters, training epochs, training data, and testing data to be the same. The data set used in this project was open sourced by Udacity for their self-driving car challenge.

The next parts of this paper are organized as follows. The next part covers the background and previous works as well as those papers on which the CNN architectures of this project are based. Section 3 consists of discussion and overview of each of the two CNN architectures used in the project. Section 4 highlights the results obtained and comparisons between all the three models. Section 5 summarizes the discussion and concludes the paper.

II. Background

Use of Neural Networks for steering angle prediction was first done by Pomerleau[4] when he introduced a small, fully-connected architecture, named Autonomous Land Vehicle in a Neural Network in 1989. The model could work in limited environment settings and obstacles. Due to limitation of hardware required for training more complex models, the use of Neural Networks for this task eventually slowed down.

In 2014, NVIDIA developed DAVE-2[1], an autonomous driving system using a simple CNN architecture as discussed before. The system consisted of a left, right and a center camera mounted on a car to record frames and the manual steering inputs corresponding to those frames. The corresponding paper showed the results and demonstrated a substantially low Root Mean Square Error, thereby demonstrating the potential of CNNs for performing this task. Furthermore, in 2017, a team of Stanford students, Andrew Simpson et al. [5], in their paper for the course CS231n tested a 3DCNN architecture with Residual connections, combined with an LSTM RNN. This hybrid architecture achieved slightly lower RMSE than the NVIDIA model. It showed that transfer learning by using pre-trained models trained on classification data sets helps in this regression task. Other approaches involving the use of Deep Reinforcement Learning[6] and Variational Autoencoders along with GANs[7] have been tested for this task with varying degrees of success.

In their paper on Squeeze-and-Excitation Networks[3], the authors proposed the use of Squeeze-and-Excitation blocks, that are placed after every convolutional block, to create a parameter corresponding to every feature-map or channel in that convolutional block. This technique when used with RestNet-154 architecture as proposed in the paper achieves state-of-the-art performance on the ImageNet classification task. Furthermore, in the YOLOv3[2] paper, the authors proposed Darknet-53, a feature extractor that is much more powerful but still more efficient than ResNet-101 or ResNet-152. In this project, we use A Modified DarkNet-53, and a Modified SE Resnet-154, which have similar overall structures, but different number of parameters and layers. Both of these architectures have approximately similar number of parameters as that of the PilotNet architecture.

III. METHODS

Two CNN architectures, apart from the PilotNet architecture as in [1], that were used on the Udacity data set are described below.

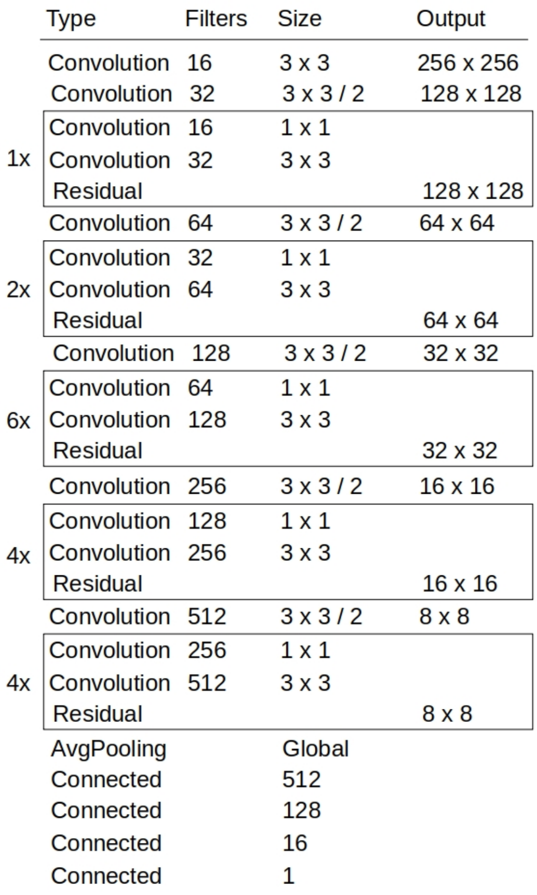
1. Modified Darknet-53:

A schematic of this architecture is given in Figure 1. The original Darknet-53 consists of approximately 44 million parameters, but this modified implementation contains approximately 9 million parameters. It consists of half the number of filters than the original in every convolutional block, and less number of repetitions of every residual block. Thus, we keep the structure to be similar to the original, but with number of parameters similar to that of PilotNet architecture, which lets us do a better comparative study between the two.

This architecture uses residual connections inside the residual blocks, wherein output of every residual block is the addition of inputs to the block and the outputs of the second convolutional layer in residual block. At the end, Global Average Pooling is used to flatten the last convolutional block, followed by 3 fully-connected layers with ReLU activation. The last fully connected layer has a single output, without any activation. This output is supposed to be a single floating point scalar in the range [-1, 1], -1 describing extreme left of the steering wheel and 1 being the extreme right, as in the labels in the data set.

2. Modified SE Resnet-154:

The SE Resnet-154, apart from having Residual connections like Darknet-53, also consists of the Squeeze-and-Excitation blocks following all the convolutional blocks. A schematic of the entire architecture hasn't been provided in [3], however, a clear textual description has been provided, which can be referred to understand the entire architecture. Furthermore, a schematic of an SE block has been provided by the authors, which are applied to all the convolutional blocks, and is as shown in Figure 2. These blocks can be thought of as being made of two steps: Squeeze, followed by Excitation. First, Global Average Pooling over a convolutional block is performed, thereby reducing the dimensions from *H x W x C* to *1 x 1 x C*. The dimensions are further reduced to *1 x 1 x* $\frac{C}{r}$ using a fully-connected layer, where r is an integer decided empirically, and *r = 16* in this case. A ReLU activation is applied on the output. These three steps are collectively known as the "Squeeze" steps. The output of ReLU layer is then applied to another fully-connected layer, followed by a Sigmoid activation, bringing back the output to size *1 x 1 x C*. The *C* scalars in this vector act as the weights corresponding to the respective *C* feature maps in the convolutional block. These steps are known as the "Excitation" steps. Finally, a channel wise multiplication of these scalars with respective feature-maps gives us the weighted convolutional block. The parameters in both the fully-connected layers are learned along with other parameters of the network. This introduces a small amount of extra computation per convolutional block, but gives a substantial boost in accuracy of most CNN architectures its applied to. A Resnet-154 with SE blocks is the current state of the art on the ImageNet classification task. Here, we use a modified version of this architecture with fewer convolutional layers and, similar to Modified Darknet, approximately same number of parameters as that of PilotNet(9 million). The final output of the the last fully-connected layer of this network is again a single scalar value in the range [-1, 1].



	Type	Filters	Size	Output
	Convolution	16	3 x 3	256 x 256
	Convolution	32	3 x 3 / 2	128 x 128
1x	Convolution	16	1 x 1	
	Convolution	32	3 x 3	
	Residual			128 x 128
	Convolution	64	3 x 3 / 2	64 x 64
2x	Convolution	32	1 x 1	
	Convolution	64	3 x 3	
	Residual			64 x 64
	Convolution	128	3 x 3 / 2	32 x 32
6x	Convolution	64	1 x 1	
	Convolution	128	3 x 3	
	Residual			32 x 32
	Convolution	256	3 x 3 / 2	16 x 16
4x	Convolution	128	1 x 1	
	Convolution	256	3 x 3	
	Residual			16 x 16
	Convolution	512	3 x 3 / 2	8 x 8
4x	Convolution	256	1 x 1	
	Convolution	512	3 x 3	
	Residual			8 x 8
	AvgPooling		Global	
	Connected		512	
	Connected		128	
	Connected		16	
	Connected		1	

Figure 1. Architecture of the Modified Darknet-53

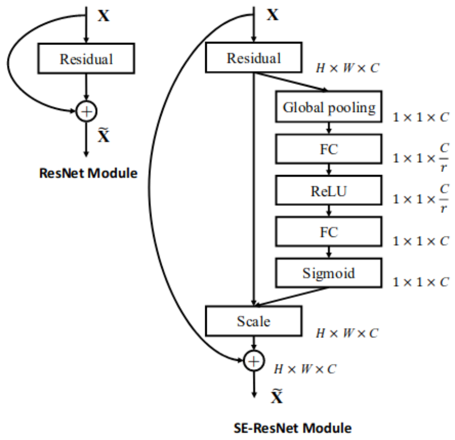


Figure 2. A Squeeze-and-Excitation block.

IV. EXPERIMENTS

Overview of the Data Set:

The data set used for this task has been collected by Udacity and open-sourced on Github[7] for anyone to use. It consists of images recorded at regular intervals from three cameras: left, center and right, facing the direction in front of the car. The steering angles applied by a human driver were also recorded corresponding to every frame captured, and were normalized in the range [-1, 1]. For our use case, we only use the frames captured from the center camera. The data consists of 33,808 image-angle pairs, out of which 27,046 were used for training and the rest for validation and testing.

Data Pre-Processing and Augmentation:

The portions of images only consisting of roads and vehicles were kept, and other irrelevant parts such as sky and front-hood of the car were cropped from the top and bottom respectively. The images were also then resized to *105 x 240* pixels, and normalized to float32 values in range [0, 1]. These pre-processed images and their corresponding angles were serialized and written to files of Tensorflow propreitary TFRecord format, for high-speed reading during training. To make sure the model doesn't starve of data while running on a GPU, a high-speed input pipeline was created using Tensorflow Data API. The data was also augmented by randomly vertically flipping images and multiplying their corresponding angles with -1, with a probability of 0.5, while reading the data for training.

Results:

Table 1. compares the Root Mean Square errors achieved by all the three models after 50 epochs of training each. Most of the training set up for all the three architecture remained the same, except for the learning rate, which was empirically adapted as needed. The training was performed on Google Colaboratory platform, which provides 12 GB of RAM, and a powerful Tesla T4 GPU with video memory of 16 GB.

As can be seen from the table, the Modified Darknet architecture with Residual connections outperforms the PilotNet architecture. Furthermore, the modified SE Resnet-154 architecture with both Residual connections as well as SE blocks outperforms the other two architectures, giving the best results. The visualization of real-time predicted angles against ground truth on some test images was performed using Comma.ai's visualization script[8] and modifying it to suite our needs. Some of the images are as shown in Figure 3.

Model name	**Number of Parameters**	**RMSE**
NVIDIA PilotNet	9,075,603	0.01286
Modified DarkNet-53	9,031,025	0.00764
Modified SE Resnet-154	9,184,720	0.00386

Table 1. RMSE Comparison between the three trained models

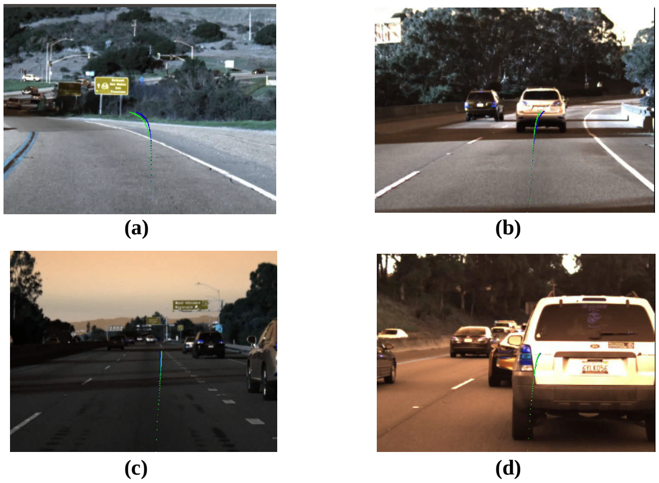

(a) **(b)** **(c)** **(d)**

Figure 3. Blue line indicates ground truth, green line indicates predicted angle. **(a)** and **(b)** show that model correctly predicts the angles on left and right turns. **(c)** shows that model correctly follows the lane on straight roads, and **(d)** shows that the model is able to correctly predict angles even when most of the road view is obstructed by the vehicles in front.

V. CONCLUSION

It can be concluded from this project that the usage of two of the newer CNN techniques: the Residual Connections, and the Squeeze-and-Excitation blocks introduced in Computer Vision improves performance on our supervised learning regression task of Steering Angle prediction. Further improvements in RMSE can be achieved by more rigorous fine-tuning and testing of previously mentioned architectures, and increasing the complexity by adding more parameters and layers, which will consequently require higher computation power. Also, by training these architectures on larger data sets of similar kind and with more diverse driving conditions, even better performance can be achieved on real

VI. REFERENCES

[1] 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. *arXiv preprint arXiv:1604.07316*, 2016.

[2] J. Redmon, A. Farhadi. YOLOv3: An Incremental Improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[3] J. Hu, L. Shen, S. Albanie, G. Sun, E. Wu: Squeeze-and-Excitation Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[4] D. A. Pomerleau. Alvinn. An autonomous land vehicle in a neural network. Technical report, Computer Science Department, Carnegie Mellon University, 1989.

[5] S. Du, H. Guo, A. Simpson. Self-Driving Car Steering Angle Prediction Based on Image Recognition. *Stanford CS231n student report*, 2017.

[6] E. Santana, G. Hotz. Learning a driving simulator. *ArXiv preprint arXiv:.1608.01230*, 2016.

[7] Udacity Self-Driving Car Challenge 2 data set: *https://github.com/udacity/self-driving-car/tree/master/datasets/CH2*

[8] Comma.ai visualization script on Github: *https://github.com/commaai/research/blob/master/view_steering_model.py*