Incremental End-to-End Learning for Lateral Control in Autonomous Driving*

JAEROCK KWON1, (Senior Member, IEEE), AWS KHALIL1, DONGHYUN KIM2, and HAEWOON NAM2 (Senior Member, IEEE)

1Department of Electrical and Computer Engineering, University of Michigan-Dearborn, Dearborn, MI 48128 USA
2Division of Electrical Engineering, Hanyang University, Ansan 15588, South Korea

Corresponding author: Jaerock Kwon (e-mail: jrkwon@umich.edu).

*This research was partially supported by the MSIT (Ministry of Science, ICT), Korea, under the High-Potential Individuals Global Training Program (2019-0-01564) supervised by the IITP (Institute for Information and Communications Technology Planning Evaluation).

ABSTRACT

The use of high-quality data is required to complete the job of lateral control utilizing Behavioral Cloning (BC) through an End-to-End (E2E) learning system. The majority of E2E learning systems gather this high-quality data all at once before beginning the training phase (i.e., the training process does not start until the end of the data collection process). The demand for high-quality data necessitates a large amount of human effort and substantial time and money spent waiting for data collection to be completed. As a result, it is critical to find a viable option to reduce both the time and cost of data collecting while also maintaining the performance of a trained vehicle controller. This paper offers a novel behavioral cloning approach for lateral vehicle control to address the aforementioned problems. The proposed technique begins by collecting the least amount of human driving data possible. The data from human drivers are utilized for training a convolutional neural network for lateral control. The trained neural network is subsequently deployed to the vehicle’s automated driving controller, replacing a human driver. At this point, a human driver is out of the loop, and an automated driving controller, trained by the initial data from a human driver, drives the vehicle to collect further training data. The driving data obtained are sent into a convolutional neural network training module, then the newly trained neural network is deployed to the automated driving controller that will drive the vehicle further. The data collection alternates neural network training processes using the collected data until the neural network learns to correctly associate an image input with a steering angle. The proposed incremental approach was extensively tested in simulated environments, and the results are promising. The incrementally trained neural networks using data collected by automated controllers were able to drive the vehicle in two different tracks successfully.

INDEX TERMS  Behavioral cloning, End-to-end learning, Artificial intelligence, Autonomous systems, Intelligent systems, Machine learning, Neural networks, Vehicle control

I. INTRODUCTION

Reinforcement Learning (RL) and Imitation Learning (IL) account for a large share of human learning. IL is also known as Apprenticeship Learning (AL) or Learning from Demonstration (LfD). Both RL and IL rely on experience. In RL, one learns to do a task based on their own experiences, which may be challenging and time-consuming. IL, on the other hand, mimics an expert’s behavior in order to learn from them [1] [2]. For example, it is unreasonable to expect a person who is given a car to learn to drive on their own without an expert’s help (RL). It would be considerably easier to show the person how to drive and have one learn from the demonstration (IL).

Similar to human learning, RL and IL are part of Machine Learning (ML) too. In RL, an agent interacts with an environment by following a policy. The agent learns by mapping states to actions. In each state of the environment, it behaves in line with the policy, receiving a reward and transitioning to a new state as a result. The goal of RL is to develop the best policy (using trial-and-error) for maximizing long-term rewards [3]. The main limitation of RL is that an optimal reward function cannot be easily found in a complex situation.

IL, on the other hand, is an alternative approach when
defining the reward function is difficult. In this paper, we provide a unique technique for Behavioral Cloning (BC), a basic type of IL, that focuses on learning the expert’s policy through supervised learning. Supervised learning is the process of learning from a training set of labeled examples given by an experienced external supervisor [3].

BC can be achieved through several methods. In this paper, the adopted BC method to perform the lateral control task is End-to-End (E2E) learning, where we train a model which maps inputs from the perception module directly to control commands (e.g., camera images to steering angle) [4]. When E2E learning is compared to traditional approaches, as illustrated in Fig. 1, the benefit of E2E learning becomes evident. In the traditional approach, the task is broken into multiple sub-tasks, which are normally completed individually (e.g., lane detection, path planning), and all important features must be carefully selected by engineers. Each sub-task has the potential to accumulate errors, resulting in an inaccurate final output. On the other hand, E2E uses data-driven approaches. There are no explicitly stated rules in the model because it is self-optimized through the training data. Better performance and less manual effort become the two key benefits of E2E learning [5].

E2E autonomous lateral control (Fig. 2) relies on high-quality driving data for deep neural network designs [6]. The majority of E2E learning systems gather this high-quality data all at once before beginning the training phase (i.e., the training process does not start until the end of the data collection process). The demand for high-quality data necessitates a large amount of human effort, and the time and money spent waiting for data collection to be completed are required [7].

As a result, it is critical to find a viable option to reduce both the time and cost of data collecting while also maintaining the performance of vehicle controllers. Our unique technique offers a practical solution in which a human drives only as much as is required, and an Artificial Intelligence (AI) chauffeur powered by a deep convolutional neural network drives on behalf of the human driver when additional driving data is required.

This paper’s contributions are the following: (1) This article proposes a unique BC process in which data collecting and neural network training are done incrementally with little initial data from a human driver. Driving data will be acquired without the need for additional human driver effort by alternating driving by a neural agent for data collection and neural network training. As a result, the high-quality training data can be collected by an AI chauffeur with bare-minimum human driving data. (2) The proposed method is built and evaluated using the open-source platform OSCAR (Open-Source Robotic Car Architecture for Research and Education), which was developed by the author. OSCAR is an open-source and full-stack robotic automobile architecture that is meant to improve robotic research and education in the context of self-driving cars. The findings of the experiments, as well as data and other material, are available online 1.

The remainder of the paper is organized as follows: Section II introduces a definition of behavior cloning based on E2E approaches and discusses data collection methodologies. Section III provides the formulation of the proposed method. Section IV presents results to demonstrate the validity of the proposed method. Finally, in Section V, the conclusions of the paper and future work are presented.

II. RELATED WORK
This section will explain the work connected to E2E learning and followed by the work related to data collecting.

A. END-TO-END LEARNING
BC through E2E learning was originally proposed and accomplished by Autonomous Land Vehicle in a Neural Network (ALVINN) [8], which demonstrated that an E2E method utilizing a fully connected neural network was capable of steering a car on a road. They employed photos from a camera and a laser range finder as input. Their neural network was shallow and at that time using a Convolutional Neural Network (CNN) was not an option.

DARPA Autonomous Vehicle (DAVE), an off-road radio control (RC) automobile, was built in a similar manner [9]. Their network learned through training examples how to extract and process important information from raw (unprocessed) video input data, such as 3-D image extraction, edge detection, object detection, and obstacle avoidance steering strategy. Instead of using a simulator, they used an RC truck. Nonetheless, behavioral cloning adopting an E2E technique did not acquire popularity until DAVE-2, which was influenced by ALVINN, was developed [5]. The original DAVE extracted essential visual information from photos captured by a front camera using a CNN. This was 10 years before CNNs demonstrated groundbreaking pattern recognition ability in the IMAGENET Large Scale Visual Recognition Challenge (ILSVRC). The DAVE-2 system offered PilotNet to scale up the original DAVE’s subscale deployment. They employed a CNN model in their research, which accepts just pictures as input and produces steering angles as output.

Bojarski et al. [10] illustrated how deep neural networks and E2E learning may be utilized to drive a vehicle. Their method produced promising outcomes and gave a clear description of how end-to-end learning works. This paper, on 1https://tinyurl.com/9j5ywvmn
the other hand, just explained and demonstrated how E2E learning works; it did not present anything new or original.

Wang et al. [11] suggested an angle branched network technique for E2E learning. As the angle branching network makes predictions based on the sub-goal angle, this strategy assisted to produce not only the steering angle but also the throttle of the automobile. The navigation instruction expressed as a one-hot vector includes less information than the sub-goal angle. They discovered that the sub-goal angle improves the driving model’s performance by improving prediction quality for the steering angle.

Wu et al. [12] presented an E2E driving model based on a Convolutional Long Short-Term Memory (Conv-LSTM) neural network with a Multi-scale Spatiotemporal Integration (MSI) module for encoding spatiotemporal input from diverse scales for steering angle prediction. They used future sequential information in the model training process to improve the spatiotemporal characteristics of the driving model for better steering angle prediction. The performance of the suggested driving model was assessed using both public Udacity data and a real-time autonomous vehicle. Despite the originality of the concept, more work is needed for smoother steering control and visualization of the proposed method.

The common process between the research papers presented is that they expect all scenarios needed for data collection, collect all data at once, then train a Neural Network (NN). In our approach, we collect the bare minimum of the required data to start training and have an automated controller drive a vehicle to collect further data in an incremental way.

**B. DATA COLLECTION**

Data collection necessitates a significant amount of human labor since E2E autonomous lateral control relies on high-quality driving data for deep neural network designs [13] [14]. Researchers in [15] explains the two phases of the data collections process using a simulator. First, they spent one month practicing the driving skills to ensure they get high quality data. The second phase also required around one month, where they run a car for 60 hours on Udacity simulator resulting in 83,424 images for model training. In [16], CARSIM simulator was used to collecting data. A human driver collected data using a joystick wheel to use it for the training step. They used 15 minutes driving video resulting in 10,800 training samples.

It is clear how much time is needed to collect a decent amount of data for autonomous driving applications. In [17], Zhang used TORCS simulator for data collection, and he collected around 390,000 samples and selected 70,000 training examples. This must have taken a lot of time. Similarly, [18], Gao collected 140,000 training images using CARLA simulator.

Table 1 shows a comparison between the traditional method, used by the above-mentioned researches, for data collection and our method for data collection. Our approach will be discussed in detail in the next sections.

| Table 1. comparing our data collection method to traditional method |
|------------------|------------------|------------------|------------------|
|                  | Simulator used  | Human driving    | Total images    | percentage of  |
|                  |                  | time             | collected       | data collected|
| [15]             | Udacity          | 60 Hour          | 83,424          | 100%           |
| [16]             | CARSIM           | 15 Mins          | 10,800          | 100%           |
| [17]             | TORCS            | NA               | 390,000         | 100%           |
| [18]             | CARLA            | NA               | 140,000         | 100%           |
| **Our method**   | OSCAR            | 1 Min            | 31,303          | 3.81%          |
Due to the amount of effort needed for data collection, many researchers tend to use public datasets. For example, in [19], they used The Berkeley DeepDrive Video dataset (BDDV), that contains videos, photos, and GPS coordinates, with boundary boxes, lane markers, and semantic labels added to the datasets [14].

There are several noteworthy public datasets. Hesai and Scale’s PandaSet contains photos as well as LiDAR information with bounding boxes and semantic labeling [20]. Waymo also offers an open dataset containing bounding boxes and semantic segmentations for pictures and LiDAR sensor data. Lyft provides open datasets created with LiDAR and camera sensors [21]. The nuScenes dataset was created with the use of LiDAR, radars, cameras, an IMU, a GPS, and annotated bounding boxes [22]. Most businesses working on autonomous cars attempt to collect as much driving data as possible since excellent datasets are critical for the development of autonomous driving systems. However, the cost of gathering and categorizing datasets is high due to the need for manual labor. This prevents most autonomous driving research communities, particularly those in academia, from gathering any form of driving data.

III. METHOD
A training data for a BC can be defined as \( D^n = \{ o^n(i), a^n(i) \}_{i=1}^T \), where \( o^n(i) \) is an observation at time \( i \), \( a^n(i) \) is its corresponding action, and \( T \) is the total time-steps. Since the proposed method uses multiple training data sets, the superscript \( n \) indicates a data identification. Learning from an expert’s demonstration \((o^n \rightarrow \text{policy} : \pi^n_0 \rightarrow a^n)\) can be depicted as in Fig. 3. The policy \( \pi^n_0 \) can be found by a Deep Neural Network (DNN) training through minimizing the prediction error of \( a^n(i) \) when an observation \( o^n(i) \) is given.

Unlike other E2E learning methods discussed in the previous section, our proposed approach to achieve BC through E2E learning has two phases. In the first and initial phase, an expert human driver collects the minimum driving data required for training: \( D^0 = \{ o^0(i), a^0(i) \}_{i=1}^T \). Then we train a DNN model to learn the policy \( \pi^0_0 \) from the expert’s demonstration. Searching the policy \( \pi^0_0 \) means learning how to map observations to actions (i.e., images to steering angles). After having the policy \( \pi^0_0 \) through a DNN training with \( D^0 \), a human driver is no longer in the loop. The distinctive point of our approach is from this point. The policy \( \pi^0_0 \) is to be deployed an automated driving controller, an AI chauffeur who will drive a vehicle to collect further driving data.

Once the policy \( \pi^0_0 \) is determined by a DNN training with \( D^0 \), the second and main phase starts. This main phase is a repeated process of collecting more data by the AI chauffeur then training another DNN model for a new and more experienced AI chauffeur. The AI chauffeur \((n = \{ 1, ..., N \}, \) where \( N \) is the total number of data collections by an AI chauffeur) will use the policy \( \pi^{n-1}_0 \). \( n = 0 \) indicates the human driver. 

Fig. 4 shows how AI drivers collect more data to increase the overall accuracy of the system. The firstly trained DNN is deployed to the 1\(^{st}\) AI chauffeur which uses the learned policy \( \pi^1_0 \) to drive a vehicle to collect more data and use the newly collected data to train a DNN; \( \pi^1_1 \) that will be deployed to the 2\(^{nd}\) AI chauffeur, and so on. The Data Collection Module in Fig. 4 represents the data collecting process of an AI chauffeur. We begin with the initial observation \( o^1(1) \) from the initial state \( s^1(1) \) and the known policy \( \pi^0_0 \). The initial observation leads to the first action \( a^1(1) \), which determines the second state \( s^1(2) \); the second state leads to the second observation \( o^1(2) \), which leads to the second action \( a^1(2) \), and so on.

Fig. 5 shows the high-level system overview. To begin, a human driver drives a virtual car and captures only the bare minimum of data needed to train a neural network to drive a short distance by replicating the human driver’s behavior. A human driver is no longer in the loop at this time. After that, a driving controller employs the trained neural network. We gave it the name of an AI chauffeur who is driving a simulated vehicle. We recorded driving data for a little longer as the AI chauffeur drove the vehicle. After training a new neural network with the newly obtained data, the AI’s brain is transplanted with a new neural network in the hopes of improving the AI’s driving performance next time. This process may be repeated iteratively until we have a neural network that can reliably predict steering angles enough to control the lateral motions of the simulated vehicle.

To implement and test our approach, we have been developing an open-source platform, OSCAR (Open-Source robotic Car Architecture for Research and education) [23]. The OSCAR platform is an open-source and full-stack robotic car architecture designed to enhance robotic research and teaching in the context of self-driving cars. Fig. 6 shows the hierarchical and modular structure of OSCAR. The Robot Operating Systems (ROS) [24] is the backend system of OSCAR. The OSCAR platform supports two vehicles: fusion and rover. The fusion is largely based on car\_demo from OSRF that was originally developed to test simulated Toyota Prius energy efficiency [25]. The backend system of
the rover is the PX4 Autopilot with ROS communicating with PX4 running on hardware or on the Gazebo simulator. In this paper, we use the fusion vehicle of the OSCAR.

A. VEHICLE DESIGN

The chassis of the car is based on the Ford Fusion model. Three cameras are mounted on the front windshield. In this article, we solely used the front camera. Ouster’s 64 channel 3D LiDAR [26] is mounted to the top of the windshield. We did not employ the LiDAR sensor for this paper. The simulated car is built as a plugin within the fusion ROS package. The vehicle’s parameters, such as maximum speed, braking torques, and maximum steering angles, may be adjusted inside the URDF (Universal Robotic Description Format) model. Fig. 7 shows the appearance of the vehicle and sensor readings.

B. DATA COLLECTION

A simulated vehicle in the OSCAR platform sends its current velocity and position in a ROS topic named/base_pose_ground_truth. The current steering angle and throttle position of the vehicle is being sent through a ROS topic named/fusion. The front camera is used to collect data in this paper. The topic name of the camera image is/fusion/front_camera/image_raw. The image message must be converted to be saved as an image file. This can be done by cv_bridge [27]. Fig. 8 shows the high-level view of the data collection system.

The steering angle range is 0 to 1 (center to left) and 0 to −1 (center to right). A driving steering wheel has 450° maximum rotation angle to the left and right direction each. This means 0 to 1 in the steering angle from the driving steering wheel is assigned to 0 to 450° and 0 to −1 is assigned to 0 to −450 degrees. We used a Logitech G920 dual-motor feedback driving force racing wheel with pedals and a gear shifter mounted on a stand (see Fig. 9).

The simulated camera has a resolution of 800 × 800. We chose the Region of Interest (ROI) as (0, 380) - (799, 530), which includes the road section front of the car omitting the sky and the front hood of the ego vehicle. The ROI will be utilized in the training of a neural network. A camera input image’s ROI will be clipped (800×151), shrunk to 160×160, and fed into the neural network. See Fig. 10 for more details.

C. NEURAL NETWORK ARCHITECTURE

We designed a convolutional neural network and named BIMINet (Bio-Inspired Machine Intelligence Network) shown in Fig. 11. The architecture of the network is inspired by PilotNet [5] where no maximum pooling layers were used. There are five convolutional layers and five fully connected layers. The first three convoluted layers have 5 × 5 filters with 2 × 2 strides. The last two convoluted layers have 3 × 3

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
The data we collected can be minimal just enough to train a neural network to be able to drive a short distance where a few straight roads and some curved roads exist. (c) neural network training system. The camera input data is fed to the neural network training module expecting to infer a steering angle associated with the input. (d) The trained neural network is deployed to the AI chauffeur. The AI chauffeur drives the vehicle as the data is being collected. The initial performance of the driving cannot be expected to be good, but the AI chauffeur will be able to drive a little bit farther compared to the previous step. After disconnecting the data flow II from a human driver, by continuing the loop, (c) $\rightarrow$ (d) $\rightarrow$ (b’) $\rightarrow$ (c), a little bit smarter neural network in the AI chauffeur will be replaced with the old one at every single loop. This incremental training and driving will eventually make the AI chauffeur’s neural network be better and better in driving.

FIGURE 6. The hierarchical and modular structure of the OSCAR. All ROS-related modules are located under the catkin_ws folder. The system settings regarding the BIMINet CNN, AI chauffeur, data collection can be done by the files under the config folder. The CNN named BIMINet is implemented with tools for training, testing, data visualization, neural network performance testing, and data analyses.

FIGURE 7. The fusion vehicle and a track of OSCAR (Open Source robotic Car Architecture for Research and education). (Left) A Ford Fusion vehicle model. (Right) Rviz visualization of sensor readings.

FIGURE 8. The high-level view of the data collection system. The Data Collection ROS node subscribes to three ROS topics: /fusion/front_camera/image_raw for camera images, /fusion for vehicle control signals, and /base_pose_ground_truth for the velocities and positions of the vehicle.

filters with $1 \times 1$ strides. The numbers of neurons in the fully connected layers are 1000, 100, 50, and 10. The total number of trainable parameters of the network is 11,054,019.

1) Training
To improve the trainability of the proposed network, we used three data preprocessing steps before using the data for training.
a: Data Preprocessing
First, data normalization was carried out. It should be noted that this is not image normalization. Because most driving circumstances involve long stretches of straight road, the data collecting module will acquire substantially more data near-zero steering angle. We must normalize the data before feeding it to the training neural network by deleting part of the data with near-zero steering degrees. Otherwise, unless well-equally distributed datasets are used, the neural network may struggle to establish the relationship between input pictures and steering angles. We established a criterion of 200 as the maximum number of data samples that might be collected. We also applied some pixel-based preprocessing. The min-max normalization is applied to the input images. Fig. 12 shows an example of data normalization.

b: Data Augmentation
We used brightness changes and affine transformation to expand the quantity of training data and produce additional variations in the datasets. The brightness of the original image was changed within the range of 50% darker and 15% brighter. When shifting occurs through affine transformation, the accompanying steering angle must be re-adjusted as well. We modify three pixels vertically and a reasonably huge 20 pixels horizontally left or right. Maximum \( \pm 0.2 \) shift around the x-axis will be applied to the steering angle. For example, if a warp affine occurs 20 pixels to the right, the steering angle will change 90° to the right.

c: Steering Angle Scaling
To improve the accuracy of steering angle prediction, we can map \(-1\) to \(1\) values to values between \(-450^\circ\) and \(450^\circ\). In an actual driving condition, just a small percentage of the steering angle data will be recorded. For this work, we used a scale factor of 5. Steering angles are scaled using...
a predetermined factor during neural network training and testing.

d: Minimal Manual Editing
Minimal manual editing is necessary at each iteration to remove poor behaviors of autonomously driven data, such as crossing left or right lanes. Identify the times when the car crashed or displayed abnormal behavior, and then remove the photographs from the acquired data. The rebuild_csv.py program that will automatically generate a CSV based on the photographs in the supplied directory can be used to reconstruct the metadata file.

e: Neural Network Training
The BIMINet CNN receives the preprocessed data. Fig. 13 depicts the whole training procedure. Before being fed into the neural network, the recorded driving data is normalized. The labeled string angle is utilized to decrease prediction errors by backpropagating to the network the difference between the desired and predicted steering angle. The training process is repeated until the validation loss does not reduce after three epochs.

3) Steering Angle Prediction in Driving
We also provide a tool by which a sequence of images are generated with information such as input image file name, labeled steering angle, predicted steering angle, the absolute value of the prediction error, vehicle velocity and position, and degree angles. Fig. 14 shows a screenshot of the data visualization.

2) AI Chauffeur
An AI chauffeur was built as a ROS node with a basic speed control technique. There is no direct control over the vehicle’s speed in the simulator. Only the throttle and brake may be used to regulate the vehicle’s speed. If we do not activate the brake, the car will eventually achieve the maximum speed since the base throttle is set to the maximum. We can picture a driver pressing the gas pedal all the way down and staying there all the while. To control longitudinal acceleration, we employed two indicators: predicted steering angle and current velocity. If the predicted steering angle is more than 0.25 (112.5°), there may be a risky bend or sharp turns ahead. When the car maintains its present speed, there is a substantial risk of going off the road. The AI chauffeur applies the brake for 1.7 seconds at 50% capacity and decreases the throttle to 10% of its maximum setting. In addition, because the training and testing tracks had twisting roads, the maximum velocity was set at 20.

We present a new metric for comparing driving performance fairly. AI chauffeurs can effectively drive the car around the entire track. However, some of them may finish the circuit with several needless lane changes and/or off-roading. To distinguish reckless driving from acceptable driving performance, we employed the autonomy meter, which assigns temporal penalties to undesirable actions such as lane violations.

\[ Autonomy = (1 - \frac{W_{inf} \times P_w + Y_{inf} \times P_y}{T}) \times 100 \]  

where \( W_{inf} \) is the number of white lane infringements, \( P_w \) is the penalty for \( W_{inf} \) in seconds, \( Y_{inf} \) is the number of yellow lane infringements, \( P_y \) is the penalty for \( Y_{inf} \) in seconds, and \( T \) is the total travel time in seconds. We used 5 for \( P_w \) and 10 for \( P_y \). This metric is inspired by the E2E learning for lane-keeping [5].
IV. EXPERIMENTAL RESULTS
To assess the suggested strategy, we used the OSCAR project, which allowed us to simulate the vehicle, sensors, and environments. OSCAR is made up of ROS packages and a convolutional neural network training/testing framework with handy utility modules that allow us to test the performance of trained neural networks, visualize activation maps, show traveled maps with distance, and create driving films from gathered datasets.

The system information where all data collection, training, and testing were taking place is as follows. The computer is Dell OptiPlex-7040 with Intel i7-6700 CPU 3.40 GHz. The size of the main memory is 32 GiB. The graphics card is NVIDIA’s GeForce GTX 745/PCLe/SSE2. The operating system is Ubuntu 18.04.5 LTS. CUDA 9 with cudnn 7.1.2.

The findings of the experiments, as well as data and other material, are available online³.

A. TRACK DESIGN
We designed two different tracks in the Gazebo world format [28] (Track A and B). Track A was utilized to collect data for neural network training. The original track design came from Dataspeed ADAS kit Gazebo/ROS [29] simulator, which includes a track for a lane-keeping demonstration. The track is created with modular road segment models that include straight road segments of 50, 100, and 200 meters and radius curve road portions of 50 and 100 meters. We embellished the track with roadside objects such as gas stations, residences, and major architectural complexes. This is important to give some variations in the roadsides to make the lane-keeping task more practical and realistic (See Fig. 15). In this paper, all data collection tasks were conducted for training in Track A.

To appropriately assess a trained neural network, a new track must be utilized in the testing phase since the neural network may recall the track where it was trained. Track B shown in Fig. 16 was created to give a different environment for testing purposes. This testing track has more curving roads with varying curvatures, a variety of straight roads, and more and larger architectural structures.

B. IMPLEMENTATION
Using the Gazebo simulator, we created ROS packages to facilitate communication between simulated vehicles and surroundings. Gazebo’s ROS integration enabled the simulation of sensors and vehicle dynamics. Two additional ROS nodes were implemented: data_collection and run_neural. The data_collection node subscribes to three ROS topics to acquire driving data from the simulated sensors and vehicle dynamics. The run_neural is an AI chauffeur by using a trained neural network to drive the simulated vehicle by publishing the ROS topic of the vehicle control such as steering, gear shift, throttle, and brake. ROS Melodic with Gazebo 9 was used for this paper.

³https://tinyurl.com/9j5ywvnm

FIGURE 15. Training road track (Track A). (Top) A bird’s-eye view of the track that is a closed-loop consisting of ten of 50-meter radius curve, six of 100-meter radius curve, two of 50-meter straight, five of 100-meter straight, three of 200-meter straight roads. The total track length is about 2,927.88 meters. This track is symmetrical at the dotted centerline. (Bottom) A profile view of the track.

We chose the Anaconda individual edition [30], which includes data science and machine learning tools as well as package management, to increase the repeatability of our work. A package environment in one system may be easily reproduced in another using Anaconda. This considerably improves a work’s reproducibility.

We used Keras 2.2.5 with TensorFlow GPU 1.12 as a back-end system to implement the BIMINet CNN. All Python libraries that are necessary for this paper can be installed by an Anaconda environment file in the main folder of the OSCAR project, oscar/config/conda/environment.yaml so that the environment can be easily created and reproduce our work.

We alternated data collection and training to evaluate the proposed method.

C. DATA COLLECTION
We began by gathering one minute of data from a human driver’s driving. A total of 1,061 camera-input photos were recorded from a human driver, as well as additional synchronized driving data with camera input such as time, throttle, velocity, and position at each point when a camera image could be received. The data collection frequency ranges between 14 and 17 Hz.

Then, using the human driving data, we trained a neural network. The steering angle predictions’ Mean Square Errors (MSE) reduced until epoch 11 (See Fig. 18 (a)). The saturated
MSE value is about 0.05, implying an average error of 0.224. Because we selected 5 as a scale factor, the real inaccuracy is 0.0448, which translates to around 20° of steering angle rotation. A 10° turn of the steering wheel (essentially a joystick) has no discernible effect on the steering angle. As a consequence, given that just a tiny dataset was utilized in the first attempt, we may interpret the outcome as not too bad. However, we must assume that steering angle prediction may not be accurate enough to manage some key places, such as strongly curved roadways.

In all, 31, 103 rows of driving data were gathered, 3, 219 data rows were removed due to the AI chauffeur’s bad behavior in data collection, and 27, 884 data rows were used. The number of data items from the human driver is 1, 061. Since the data collection frequency is about 14 to 17 Hz, it took roughly one minute to acquire the initial data to start training a neural network. Only 3.81% (1, 061 out of 27, 884) of the total data came from a human driver. The AI chauffeur was used to collect seven data sets in an incremental way after one minute of human driving. See Fig. 17 for more details.

In the trial when we attempted to collect two-minute data, the AI chauffeur could only drive for one minute and 42 seconds (labeled 1m42s). After eliminating 140 data rows that displayed abnormal behavior, we utilized the 1m42s data to train a neural network again to gather two-minute data. The newly trained network was then deployed to the AI chauffeur, who could drive for two minutes. We repeated this alternation between data collection and training with the collected data without a human driver in the loop. Up to eight minutes, data collection was conducted because it takes about eight minutes to completely drive the road track for training.

**D. TRAINING AND TESTING IN TRACK A**

For predicting steering angles, we employed Mean Square Error (MSE) as the training loss function. In the gradient descent, the Adam optimizer with a learning rate of 0.001 was utilized. The training to validation ratio was 7 : 3. The batch size was sixteen. We implemented a callback that monitors validation loss at an early stage. If it falls in three consecutive epochs, the training is terminated. In each batch, the rows of input data were shuffled.

The training graphs are shown in Fig. 18 where we can see training in eight cases was successfully conducted. Just note that the scale of the y axis is different from each training. As an example, Fig. 18 (a) appears to be better than Fig. 18 (h), but the scale in (a) is tenfold of (h).

Fig. 19 is the scatter plots for ground truth values vs. predictions for steering angles. In general, all graphs show highly positive correlations between labels and predictions so that we can tell that all trained neural networks have pretty decent prediction performance. If we take a look at the graphs more carefully, we can tell (g) and (h) are better than (a) - (e) in Fig. 19 since the narrower plot shows smaller prediction errors. The results are reasonable considering that longer quality datasets are better for training a neural network.

Actual steering angle values from the neural network predictions are shown in Fig. 20 with ground truth values. As we can see in the graph, only small errors are reported. Fig. 21 shows that MAE decreases as longer minute datasets were used.

We also report the traveled distance of the vehicle on Track A map during data collection in Fig. 22.

As Fig. 18 - Fig. 22 show, we were able to train a neural
network for the whole track with only one minute of driving data from a human driver. To summarize, the procedures are as follows. The human driver was removed from the loop after we gathered the initial training data. The first trained neural network was dispatched to be utilized as a driving controller for an AI chauffeur who could drive for one minute and 42 seconds while we collected the AI chauffeur’s driving data. We trained a neural network labeled 1m42s on the obtained data from the AI chauffeur and delivered it to an AI chauffeur to drive a two-minute trip while collecting chauffeur driving data. The obtained two-minute driving data was used to train a new neural network. These two phases, data collection and training a new neural network, were repeated until the AI chauffeur could drive the whole track in Track A.

E. TESTING IN TRACK B

We created another track for testing to see if the proposed approach could be driven in a new track (Track B). This track contains a variety of curves with a variety of ambient elements to verify trained neural networks’ ability to predict steering angles correctly even on a different course.

TABLE 2. Two conditions for testing in a different track

| Driving Strategy                  | Condition 1 | Condition 2 |
|-----------------------------------|-------------|-------------|
| Maximum Velocity                  | 20 m/h      | 18 m/h      |
| Sharp Curve Criterion             | 112.5 degrees | 90 degrees |
| Brake Application Time            | 1.7 seconds | 1.9 seconds |
| Brake Capacity                    | 50%         | 10%         |
| Throttle in Sharp Curve           | 10%         | 10%         |
| Default Throttle                  | 100%        | 100%        |

We designed two different driving conditions for testing. The detailed values of driving strategies can be found in Table 2. Under Condition 1, we had the AI chauffeur who has the 8m neural network to drive Track B. The AI chauffeur was able to drive the vehicle on Track B for more than 70% of the track even though the AI chauffeur...
FIGURE 20. Comparison of labeled steering angles and ground truth values. (a) - (h) are one through eight minutes respectively. The blue is the ground truth and the orange is the prediction. The x axis is the time step. The y axis is the steering angle. Only slight differences can be witnessed and MAE (Mean Absolute Error) values are also small and decrease as longer minute data are used.

FIGURE 21. The Mean Absolute Error (MAE) decreases as longer minute datasets that are used in a neural network training. The slope of the graph indicates that the prediction accuracy increases as more datasets are used in training.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
FIGURE 23. Driven distances during data collection on Track B under Condition 1. The $x$ and $y$ axes are for the areas of the track in meters. (a) - (h) are one through eight minutes respectively. Note that the scales of (a)-(e) are different from (f)-(h). The bottom left point (0, 0) is the starting location and the initial orientation of the vehicle is to the right side.

FIGURE 24. Driving performance in the autonomy metric under Condition 1. The AI chauffeur who has the 8m neural network completes the track over 70%.

FIGURE 25. Driven distances during data collection on Track B under Condition 2. The $x$ and $y$ axes are for the areas of the track in meters. (a) - (h) are one through eight minutes respectively. Note that the scales of (a) and (c) are different from (b), (d)-(h). The bottom left point (0, 0) is the starting location and the initial orientation of the vehicle is to the right side.

FIGURE 26. Driving performance in the autonomy metric under Condition 2. From 6m, the trained neural networks show higher than 95% autonomy.
FIGURE 27. Activation maps. (a) - (i) are one minute, one minute 42 seconds, two to eight minutes respectively. Left in each label is a camera image input and right is an overlaid activation map on the region of interest. The blue is low and red is high activation values. The activations are taken from the last convolutional layer in the network. (a) show low confidence overall but some activations on the edges of the road. Interestingly up to five-minute data (d)-(f), neural networks learned the shapes of the roads is more important than the road itself. This can be a reasonable choice for neural networks to minimize MSEs of predictions due to the lack of varieties on both sides of the road. As more datasets were fed to the neural network training (g)-(i), the neural networks have to change the strategy to minimize MSEs due to different objects residing on the left and/or right side of the road that are not supposed to affect the steering angle predictions.

TABLE 3. Driving performance in the autonomy metric under Condition 1. The total length of the track is 2,927.88 m. Autonomy are calculated with (1).

| Time | Distance Traveled | Completion | White Inf. | Yellow Inf. | Autonomy |
|------|-------------------|------------|------------|-------------|----------|
| 1m   | 309.79            | 10.58%     | 1          | 2           | 9.04%    |
| 1m42s| 159.50            | 5.45%      | 0          | 1           | 4.76%    |
| 2m   | 173.26            | 5.92%      | 1          | 1           | 5.06%    |
| 3m   | 528.03            | 18.03%     | 2          | 7           | 12.91%   |
| 4m   | 151.97            | 5.19%      | 0          | 1           | 4.51%    |
| 5m   | 1,028.31          | 35.12%     | 1          | 7           | 30.17%   |
| 6m   | 1,031.25          | 35.22%     | 1          | 7           | 30.27%   |
| 7m   | 409.40            | 13.98%     | 0          | 4           | 11.25%   |
| 8m   | 2,190.79          | 74.83%     | 1          | 8           | 71.92%   |

TABLE 4. Driving performance in the autonomy metric under Condition 2. The total length of the track is 2,278.10 m. Autonomy are calculated with (1).

| Time | Distance Traveled | Completion | White Inf. | Yellow Inf. | Autonomy |
|------|-------------------|------------|------------|-------------|----------|
| 1m   | 354.67            | 15.57%     | 2          | 2           | 13.37%   |
| 1m42s| 417.99            | 18.35%     | 2          | 5           | 13.52%   |
| 2m   | 1,927.70          | 84.62%     | 9          | 7           | 76.50%   |
| 3m   | 528.03            | 23.18%     | 0          | 6           | 17.91%   |
| 4m   | 2,179.37          | 95.67%     | 2          | 10          | 96.43%   |
| 5m   | 2,278.10          | 100.00%    | 2          | 6           | 96.27%   |
| 6m   | 2,278.10          | 100.00%    | 1          | 4           | 96.27%   |
| 7m   | 2,278.10          | 100.00%    | 1          | 2           | 98.02%   |
| 8m   | 2,278.10          | 100.00%    | 1          | 5           | 95.39%   |

V. CONCLUSION

In this paper, we propose a unique BC process in which data collecting and neural network training are done incrementally with little initial data from a human driver. Only 3.81% (1,061 out of 27,884) of the total data came from a human driver. The AI chauffeur was used to collect seven data sets in an incremental way after approximately one minute of human driving. The training graphs showed that training in eight cases was successfully conducted. The scatter plots for ground truth values contrasted to predictions for steering angles showed highly positive correlations between labels and predictions so that we can tell that all trained neural networks have pretty decent prediction performance. Extensive testing using two separate tracks and two different driving circumstances successfully validated the proposed method.
We also developed OSCAR, an open-source platform for autonomous driving to conduct experiments in a systematic way. We anticipate that by utilizing this incremental driving data collection and neural network controller training with the open-source platform, OSCAR, significant human effort and time will be saved, and the development process of autonomous car technology will be expedited. As future work, we are planning to automate the entire process except for the initial human driving data.

ACKNOWLEDGMENT
Special thanks to Bada Kwon for his efforts in data collection and neural network training. All collected data and post-processing of the data were done by Bada Kwon. Also, he trained nine different neural networks and ran them on a new test track built by himself.

REFERENCES
[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In Proceedings of the twenty-first international conference on Machine learning, page 1, 2004.
[2] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. arXiv preprint arXiv:1805.01954, 2018.
[3] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. MIT press, 2018.
[4] Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 4693–4700, 2018.
[5] Z. Chen and X. Huang. End-to-end learning for lane keeping of self-driving cars. In 2017 IEEE Intelligent Vehicles Symposium (IV), pages 1856–1860, June 2017.
[6] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to End Learning for Self-Driving Cars. arXiv:1604.07316 [cs], April 2016. arXiv: 1604.07316.
[7] Siddharth Agarwal, Ankit Vora, Gaurav Pandey, Wayne Williams, Helen Kourousi, and James McBride. Ford Multi-AV Seasonal Dataset. The International Journal of Robotics Research, 39(12):1367–1376, October 2020. Publisher: SAGE Publications Ltd STM.
[8] Dean Pomerleau. ALVINN: An Autonomous Land Vehicle In a Neural Network. In Proceedings of Advances in Neural Information Processing Systems 1, pages 305–313. Morgan Kaufmann, 1989.
[9] Net-Scale. Autonomous Off-Road Vehicle Control Using End-to-End Learning, July 2004.
[10] Mariusz Bojański, Philip Yeres, Anna Choromanska, Krzysztof Choromanski, Bernhard Firner, Lawrence Jackel, and Urs Muller. Explaining how a deep neural network trained with end-to-end learning steers a car, 2017.
[11] Qing Wang, Long Chen, Bin Tian, Wei Tian, Lingxi Li, and Dongpu Cao. End-to-End Autonomous Driving: An Angle Branched Network Approach. IEEE Transactions on Vehicular Technology, 68(12):11599–11610, December 2019. Conference Name: IEEE Transactions on Vehicular Technology.
[12] Tianhao Wu, Ao Luo, Rui Huang, Hong Cheng, and Yang Zhao. End-to-End Driving Model for Steering Control of Autonomous Vehicles with Future Spatiotemporal Features. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 950–955, November 2019. ISSN: 2153-0866.
[13] S. Sharma, G. Tewolde, and J. Kwon. Lateral and Longitudinal Motion Control of Autonomous Vehicles using Deep Learning. In 2019 IEEE International Conference on Electro Information Technology (EIT), pages 1–5, May 2019. ISSN: 2154-0373.
[14] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell. BDD100K. A Diverse Driving Dataset for Heterogeneous Multitask Learning. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 2633–2642, June 2020. ISSN: 2575-7075.
[15] Yaqin Wang, Dongfang Liu, Hyewon Jeon, Zhiwei Chu, and Eric Matson. End-to-end Learning Approach for Autonomous Driving: A Convolutional Neural Network Model. In Proceedings of the 11th International Conference on Agents and Artificial Intelligence, pages 833–839, Prague, Czech Republic, 2019. SCITEPRESS - Science and Technology Publications.
[16] Viktor Rausch, Andreas Hansen, Eugen Solowjow, Chang Liu, Edwin Kreuzer, and J. Karl Hedrick. Learning a deep neural net policy for end-to-end control of autonomous driving. In 2017 American Control Conference (ACC), pages 4914–4919, 2017.
[17] J. Zhang. End-to-end Learning for Autonomous Driving, PhD thesis, Dept. of Comp. Sci., New York Univ., New York City, NY, USA, 2019.
[18] Y. Gao. End to End Learning in Autonomous Driving Systems. PhD thesis, Dept. of Comp. Sci., University of California., Berkeley, CA, USA, 2019.
[19] Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. End-to-End Learning of Driving Models from Large-Scale Video Datasets. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3530–3538, Honolulu, HI, July 2017. IEEE.
[20] Scale. PandaSet Open Datasets - Scale.
[21] Waymo. Waymo Open Dataset.
[22] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A Multimodal Dataset for Autonomous Driving. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 11618–11628, Seattle, WA, USA, June 2020. IEEE.
[23] Jaerock Kwon. OSCAR (Open-Source robotic Car Architecture for Research and education), February 2021. original-date: 2020-11-29T19:48:57Z.
[24] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tullio Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. ROS: an open-source Robot Operating System. In Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics, Kobe, Japan, May 2009.
[25] Open Robotics Foundation. Demo of Prius in ROS/GAZEBO, February 2021. original-date: 2017-06-20T23:05:01Z.
[26] Ouster. About Ouster - a team of experts in lidar, optics, and perception.
[27] Open Robotics Foundation. Interface OpenCV with ROS, cv_bridge.
[28] Gazebo. Gazebo : Tutorial : Building a world.
[29] DataSpeed. DataspeedInc / dataspeed_dbw_simulation — Bitbucket.
[30] Anaconda. Anaconda | Individual Edition.
JAEROCK KWON (M’06–SM’20) received his B.S. and M.S. degrees from Hanyang University, Seoul, Korea, in 1992 and 1994, respectively. Between 1994 and 2004, he worked for LG Electronics, SK Teletech, and Qualcomm Internet Service. Then, he received his Ph.D. degree in Computer Engineering from Texas A&M University, College Station, USA, in 2009.

From 2009 to 2010, he was a professor of the Department of Electrical and Computer Engineering at Kettering University, Flint, MI, USA. From 2010, he has been a professor of the Department of Electrical and Computer Engineering at the University of Michigan-Dearborn, MI, USA. His research interests include mobile robotics, autonomous vehicles, and artificial intelligence.

Dr. Kwon’s awards and honors include the Outstanding Researcher Award, Faculty Research Fellowship (Kettering University), and SK Excellent Employee (SK Teletech). He serves as President of the Korean Computer Scientists and Engineers Association in America (KOCSEA) in 2020 and 2021.

AWS KHALIL earned a B.S. in Mechatronics Engineering from the University of Jordan, Amman, Jordan in 2018. He received his M.S. degree from Budapest University of Technology and Economics (BME) in Autonomous Vehicle Control Engineering. He is currently a Ph.D. student in Electrical, Electronics, and Computer Engineering at the University of Michigan - Dearborn. Mr. Khalil is interested in self-driving cars. During his master’s studies, he worked as a research assistant at BME Automated Drive Lab, and now he works at the University of Michigan - Dearborn’s Bio-inspired Machine Intelligence (BIMI) Laboratory.

HAEWOON NAM (S’99–M’07–SM’10) received the B.S. degree from Hanyang University, Seoul, South Korea, the M.S. degree from Seoul National University, and the Ph.D. degree in electrical and computer engineering from The University of Texas at Austin, Austin, TX, USA. From 1999 to 2002, he was with Samsung Electronics, Suwon, South Korea, where he was engaged in the design and development of code division multiple access and global system for mobile communications (GSM)/general packet radio service baseband modem processors. In the summer of 2003, he was with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA, where he performed extensive radio channel measurements and analysis at 60 GHz. In the fall of 2005, he was with the Wireless Mobile System Group, Freescale Semiconductor, Austin, where he was engaged in the design and test of the worldwide interoperability for microwave access (WiMAX) medium access control layer. His industry experience also includes work with the Samsung Advanced Institute of Technology, Kiheung, South Korea, where he participated in the simulation of multi-input–multi-output systems for the Third-Generation Partnership Project (3GPP) Long-Term Evolution (LTE) standard. In October 2006, he joined the Mobile Devices Technology Office, Motorola, Inc., Austin, where he was involved in algorithm design and development for the 3GPP LTE mobile systems, including modeling of 3GPP LTE modem processors. Later in 2010, he was with Apple Inc., Cupertino, CA, USA, where he worked on research and development of next-generation smart mobile systems. Since March 2011, he has been with the Division of Electrical Engineering, Hanyang University, Ansan, South Korea, where he is currently a Professor. He received the Korean Government Overseas Scholarship for his doctoral studies in the field of electrical engineering.

DONGHYUN KIM received the B.S. degree in electronics and communication engineering from Hanyang University, Ansan, South Korea, in 2018, where he is currently pursuing a Ph.D. degree. His research interests include deep learning-based autonomous driving, personalized Adaptive Cruise Control. He received numerous scholarships, such as the Korean Government Scholarship for his M.S. and Ph.D. degrees.