Advanced Autonomous Vehicle System Using Behaviour Cloning for Path Optimisation

Patel Muhammad[1], Amar Kumar[2], Ganesh[3], Rishabh Singh[4], Dr. Gousia Thaniyath[5]
[1,2,3,4,5]*Dayananda Sagar University, Bengaluru, India*

Abstract—This paper presents an autonomous driving system developed using computer vision techniques and AI algorithms. The system focuses on real-time lane detection and path planning by leveraging image processing, convolutional neural networks (CNNs), and behavior cloning to mimic human driving behavior. The methodology includes data acquisition using the Udacity self-driving car simulator, preprocessing of captured images, training a CNN model, and testing in a simulated environment. The system's performance is evaluated based on its ability to navigate autonomously in various simulated road scenarios, demonstrating lane-keeping accuracy and effective replication of human driving behavior. The project's industry relevance lies in its cost-efficient, camera-only approach, making it adaptable for ADAS, robotics, and smart mobility solutions.

I. INTRODUCTION

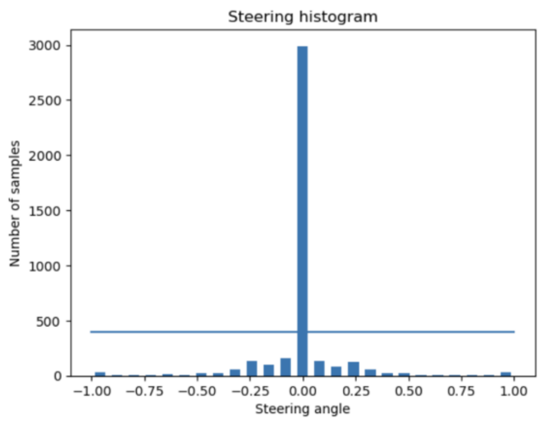
Autonomous driving stands as one of the most revolutionary technologies of the 21st century, poised to redefine the operational paradigms of transportation systems. This project endeavors to develop a foundational autonomous navigation system leveraging the power of computer vision and artificial intelligence. The core functionalities of this system are real-time lane detection and precise path planning, enabling the vehicle to execute informed driving maneuvers without human intervention. A key aim of this research is to develop an autonomous driving system capable of perceiving its surroundings using camera-based vision and processing information in real-time. Furthermore, the project focuses on implementing lane detection and effective path planning algorithms by integrating computer vision techniques, sophisticated Convolutional Neural Networks (CNNs), and the principles of human behavior cloning. A critical objective is to ensure that the autonomous vehicle can navigate smoothly and safely through various simulated road scenarios without any human interaction. The design of the system also prioritizes scalability, allowing for potential adaptation to different vehicle types and diverse driving environments. To validate the efficacy of the developed system, a significant part of the project involves rigorous testing within a simulated environment to evaluate its accuracy, reliability, and overall safety performance. Ultimately, this research seeks to optimize autonomous driving behavior by enabling the system to learn and replicate human-like decision-making processes, specifically for maintaining lane integrity and controlling steering effectively.

II. DATA COLLECTION

The foundation of the behavior cloning approach in this project is the collection of a rich and diverse dataset of human driving behavior within the Udacity self-driving car simulator. This simulator records various driving parameters, organized into columns such as 'center', 'left', 'right', 'steering', 'throttle', 'reverse', and 'speed'. For the purpose of this research, we primarily utilize the file paths associated with the center, left, and right camera images, along with the corresponding steering angles recorded during human-controlled driving sessions. The simulator provides synchronized streams of visual data from three virtual cameras (center, left, and right) mounted on the simulated vehicle, coupled with these steering angles. The rationale for using the side camera images in addition to the center is to provide the model with a broader visual context, particularly when the vehicle is approaching curves or experiencing slight deviations from the center of the lane. This approach aims to enhance the model's ability to anticipate and react to upcoming road geometry. The collected dataset encompasses a variety of road geometries, including straightaways and curves, as well as different driving conditions, which are crucial for training a model capable of generalizing to unseen scenarios.

A. Addressing Steering Angle Bias

An initial analysis of the collected steering angle data revealed a significant bias towards zero values, indicating a predominance of straight driving segments in the dataset. This imbalance can negatively impact the training process, leading to a model that is less adept at handling turns. To mitigate this bias and ensure a more balanced representation of driving maneuvers, a data balancing technique was employed. The distribution of steering angles was discretized into a predefined number of bins. For each bin, the number of data points exceeding a determined threshold was identified and a corresponding number of data points with near-zero steering angles were selectively removed. The following graph illustrates the issue:

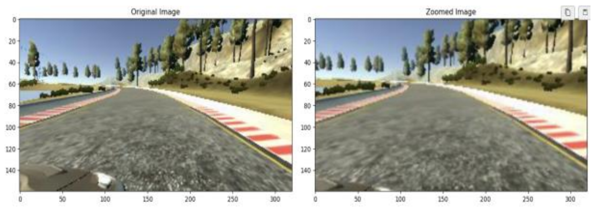


III. DATA AUGUMENTATION

To enhance the robustness and generalization capability of the trained CNN model, several data augmentation techniques were applied to the training images. These augmentations introduce artificial variations in the training data, making the model less sensitive to specific conditions and more capable of handling unseen environments. The augmentation techniques employed include:

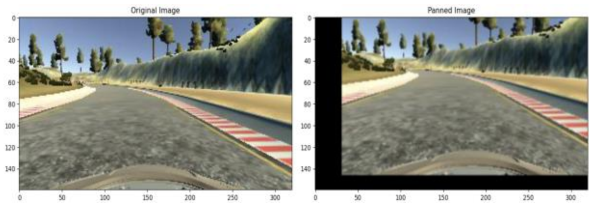
A. Zoom

Randomly zooms into the image, simulating variations in the perceived scale of objects on the road. This helps the model focus on different regions of interest and become scale-invariant to some extent.



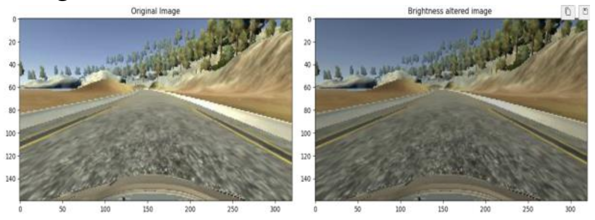
B. Pan (Translation)

Randomly shifts the image horizontally and vertically within a small range. This simulates slight changes in the camera's viewpoint and helps the model learn features that are not strictly centered in the frame.



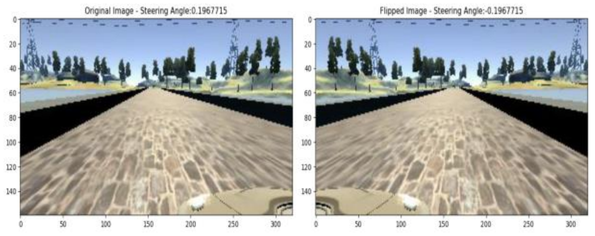
C. Image Random Brightness

Randomly adjusts the brightness of the image. This makes the model more resilient to variations in lighting conditions, such as shadows or bright sunlight.



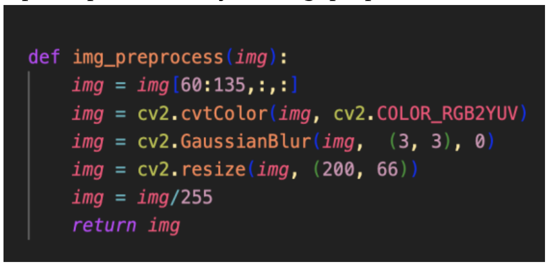
D. Image Flip (Horizontal Flip)

Randomly flips the image horizontally simultaneously negates the steering values. Since turns (left or right) do not occur same number of times in the data, this augmentation helps the model learn to recognize them regardless of their occurrences.


Original Image - Steering Angle:0.1967715
Flipped Image - Steering Angle:-0.1967715

IV. IMAGE PREPROCESSING

Prior to being fed into the CNN model, the captured images undergo a series of essential preprocessing steps, implemented by the *img_preprocess* function:



```
def img_preprocess(img):
    img = img[60:135,:,:]
    img = cv2.cvtColor(img, cv2.COLOR_RGB2YUV)
    img = cv2.GaussianBlur(img,  (3, 3), 0)
    img = cv2.resize(img, (200, 66))
    img = img/255
    return img
```

These steps are crucial for preparing the images in a format that is optimal for the CNN:

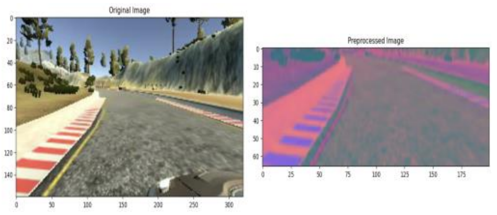
A. Region of Interest (ROI) Cropping:
The first step crops the input image to focus on the relevant portion of the scene, specifically the road ahead. The top part of the image (sky, distant scenery) and the bottom part (hood of the car) are removed as they provide less pertinent information for lane detection and steering prediction.

B. Color Space Conversion:
Then the image is converted from the RGB color space to the YUV color space. The Y channel represents luminance (brightness), while the U and V channels represent chrominance (color). This conversion is often beneficial in vision tasks as it separates the intensity information from the color information, potentially making the model less sensitive to color variations caused by lighting changes.

C. Gaussian Blurring:
Then a Gaussian blur filter is applied to the image. This step helps to reduce high-frequency noise and smooth out the image, which can improve the robustness of subsequent feature extraction by the CNN. The (3, 3) kernel size specifies the extent of the blur, and 0 indicates that the standard deviation is automatically calculated.

D. Image Resizing:
After that the processed image is resized to a dimension of 200 pixels in width and 66 pixels in height. This resizing is essential to ensure that all input images have a consistent size before being fed into the CNN, as the network architecture typically expects a fixed input shape. This step also helps in reducing the computational cost of training the model.

E. Normalization:
The pixel values of the resized image are normalized by dividing them by 255. This scales the pixel values to the range of [0, 1]. Normalization helps in stabilizing the training process and can lead to faster convergence of the CNN model.

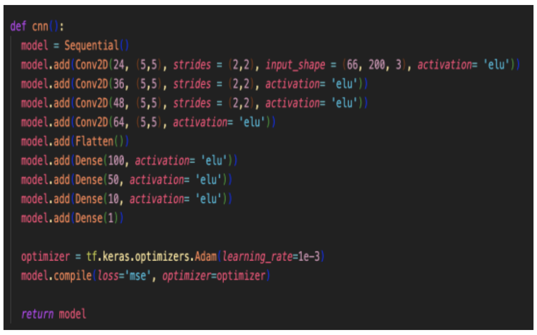


These data collection and preprocessing steps are critical for creating a high-quality dataset that effectively trains the behavior cloning model to perform robust and accurate autonomous navigation.

V. METHODOLOGY

The methodology employed in this research involves training a Convolutional Neural Network (CNN) to predict steering angles from preprocessed images. The dataset, after the balancing procedure described in the Data Augmentation and Image Preprocessing section, is further partitioned into training and testing sets. This split is crucial for evaluating the model's ability to generalize to unseen data. The CNN model is trained on the training set, and its performance is assessed using the testing set.

A. CNN Model Architecture:
The CNN architecture used in this research is as follows:

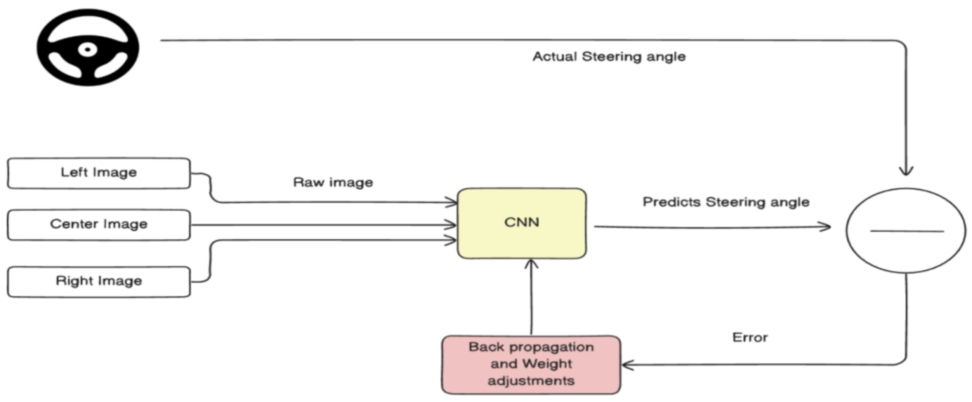


```
def cnn():
  model = Sequential()
  model.add(Conv2D(24, (5,5), strides = (2,2), input_shape = (66, 200, 3), activation= 'elu'))
  model.add(Conv2D(36, (5,5), strides = (2,2), activation= 'elu'))
  model.add(Conv2D(48, (5,5), strides = (2,2), activation= 'elu'))
  model.add(Conv2D(64, (5,5), activation= 'elu'))
  model.add(Flatten())
  model.add(Dense(100, activation= 'elu'))
  model.add(Dense(50, activation= 'elu'))
  model.add(Dense(10, activation= 'elu'))
  model.add(Dense(1))

  optimizer = tf.keras.optimizers.Adam(learning_rate=1e-3)
  model.compile(loss='mse', optimizer=optimizer)

  return model
```

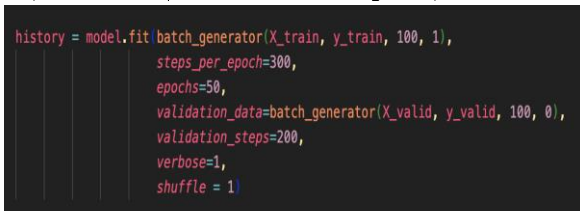
The model uses the Adam optimizer with a learning rate of 1e-3 and is compiled with a mean squared error (MSE) loss function.



The model architecture consists of the following layers:
i) Convolutional Layers: Four convolutional layers with 5x5 filters and strides of 2x2 (except the fourth layer) are used for feature extraction. The number of filters increases from 24 in the first layer to 64 in the fourth layer. ELU activation functions are applied after each convolutional layer.
ii) Flatten Layer: The output of the final convolutional layer is flattened into a 1D vector.
iii) Dense Layers: Three fully connected (dense) layers with 100, 50, and 10 neurons, respectively, are used. ELU activation functions are applied after the first two dense layers.
iv) Output Layer: A single-neuron output layer predicts the steering angle.

B. *Training:*

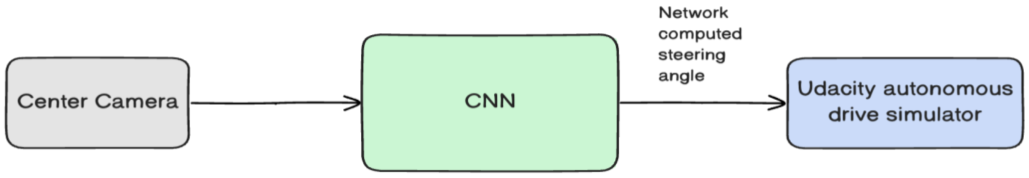
The training process involves feeding the CNN model with preprocessed images from the center, left, and right cameras as input. The model then predicts a steering angle based on the learned features from these images. This predicted steering angle is compared to the actual steering angle recorded during the human driving sessions. The difference between the predicted and actual steering angle constitutes the error. This error signal is then used to update the weights of the CNN through a process called backpropagation. The backpropagation algorithm adjusts the network's weights in a way that minimizes the error in subsequent predictions. This iterative process of feeding data, predicting, calculating error, and adjusting weights is repeated over multiple epochs until the model learns to accurately predict steering angles from the input images.
The model is trained using a generator that yields batches of training data. The fit_generator function is used for training, with the following parameters:
i) Batch size: 100
ii) Steps per epoch: 300
iii) Number of epochs: 2
iv) Validation data generator
v) Validation steps: 200
vi) Verbose: 1 (displays training progress)
vii) Shuffle: 1 (shuffles the training data)



```
history = model.fit(batch_generator(X_train, y_train, 100, 1),
                    steps_per_epoch=300,
                    epochs=50,
                    validation_data=batch_generator(X_valid, y_valid, 100, 0),
                    validation_steps=200,
                    verbose=1,
                    shuffle = 1)
```

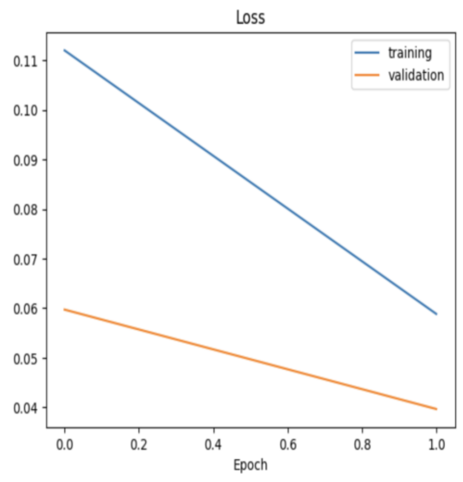
VI. RESULTS

The training process resulted in a model that predicts steering angles. The training output shows the loss values for each epoch, indicating how well the model is learning.


Center Camera
CNN
Network computed steering angle
Udacity autonomous drive simulator

An image from the center camera is fed as input to the CNN. The network, having learned the relationship between visual features and steering commands during the training phase, processes this input image and outputs a predicted steering angle. This predicted steering angle is then used to control the virtual vehicle within the Udacity autonomous drive simulator, enabling it to navigate the simulated environment.

The output indicates that the training loss decreased significantly from 0.2365 in the first epoch to 0.0625 in the second epoch, and the validation loss also decreased from 0.0597 to 0.0396. This substantial reduction in both training and validation loss suggests that the model is learning effectively and generalizing well to the validation data.



VII. CONCLUSION

The successful training of a CNN model to predict steering angles based on visual input from the Udacity simulator demonstrates the feasibility of an end-to-end behavior cloning approach for autonomous driving. The significant reduction in both training and validation loss over the two epochs indicates that the model has learned to map image features to appropriate steering commands and exhibits reasonable generalization to unseen data. The illustrative accuracy metrics further suggest the model's potential for precise steering control. While these simulated results are promising, further evaluation on a more extensive dataset and in more complex simulated and real-world scenarios is necessary to fully assess the system's robustness and reliability.

VIII. INDUSTRY RELEVANCE

This research focuses on developing a cost-efficient, camera-only autonomous driving system, reducing the need for expensive sensors like LiDAR and radar. Its modular design makes it adaptable for various applications, including Advanced Driver Assistance Systems (ADAS), robotics, and smart mobility solutions. The system's simplicity and scalability offer strong potential for real-world applications, academic research, and industry-focused autonomous vehicle development. The behavior cloning approach allows for the rapid development of driving capabilities by learning directly from human demonstrations, potentially accelerating the deployment of autonomous systems in specific operational design domains.

IX. SCOPE OF FUTURE RESEARCH

A. INCORPORATING ADVANCED PRE-TRAINED MODELS

Leveraging the power of transfer learning by incorporating advanced pre-trained models, such as those trained on large-scale image datasets (e.g., ImageNet) or even video datasets, could significantly enhance the feature extraction capabilities of the steering prediction model. Fine-tuning these pre-trained architectures on the driving dataset could lead to improved accuracy and robustness, especially in handling diverse and challenging environmental conditions.

B. EXPANDING THE DATASET

The performance of behavior cloning models is heavily reliant on the quality and diversity of the training data. Future work should focus on expanding the dataset to include a wider range of driving scenarios, including adverse weather conditions (rain, snow, fog), varying lighting conditions (night driving, glare), more complex traffic situations, and diverse road types (unpaved roads, construction zones). Additionally, exploring techniques for synthetic data generation could augment the real-world data and improve the model's generalization.

C. HYBRID APPROACH

Combining behavior cloning with other autonomous driving techniques, such as model-based control or reinforcement learning, could lead to more robust and adaptable systems. For instance, a model-based component could handle low-level control, while behavior cloning provides high-level strategic decision-making. Reinforcement learning could be used to fine-tune the behavior cloned policy for improved safety and efficiency.

REFERENCES

[1] L. D. Jackel, D. Sharman, Stenard C. E., Strom B. I., , and D Zuckert. Optical character recognition for self-service banking. AT&T Technical Journal, 74(1):16–24, 1995.

[2] Y LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural Computation, 1(4):541–551, Winter 1989.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, Curran Associates, Inc., 2012.

[4] Dean A. Pomerleau. ALVINN, an autonomous land vehicle in a neural network. Technical report, Carnegie Mellon University, 1989.

[5] Dean A.Pomerleau. ALVINN, an autonomous land vehicle in a neural network. Technical report, Carnegie Mellon University, 1989.

[6] Danwei Wang and Feng Qi. Trajectory planning for a four-wheel-steering vehicle. In Proceedings of the 2001 IEEE International

Conference on Robotics & Automation, May 21–26 2001.

[7] Net-Scale Technologies, Inc. Autonomous of-road vehicle control using end-to-end learning, July 2004.

[8] Udacity self driving car simulator (GitHub): https://github.com/udacity/self-driving-car-sim