Implementation of an Artificial Intelligence Model in a Self-Driving Vehicle Using Deep Learning

Xiyana V. Figuera

Capstone Design Project — Department of Convergence Mechanical Engineering, Silla University, Busan, South Korea

Advisor: Prof. Ki-Hyun Kim · 2021

Abstract—This project implements a vision-based self-driving model on a small physical robot car that learns to follow a track by imitation (behavioral cloning). A camera frame is mapped to a driving action with a convolutional neural network trained on data recorded while driving the car manually. Starting from the well-known NVIDIA PilotNet model, which is designed for large datasets and continuous-steering regression, the model is adapted for a small, self-collected dataset (≈1,900 frames): the steering task is reframed as a three-class action classification, the mean-squared-error objective is replaced by a categorical negative-log-likelihood loss, ReLU is replaced by PReLU, and the convolutional stack is made lighter with a wider first kernel. These changes raised validation accuracy from 57% to 81% and let the car complete the track reliably.

1. INTRODUCTION

Self-driving is commonly approached with supervised learning (as in early Tesla Autopilot) or reinforcement learning (as in Waymo's large-scale simulation). Both rely on very large amounts of data or simulation, which is impractical for an individual student project. The goal of this work is to build an intelligent car that learns to drive on a physical track from a *small* dataset collected by the author, using computer-vision preprocessing and a compact deep-learning model. Beyond robotics, methods that learn well from small datasets are valuable in any data-scarce domain.

The approach is *behavioral cloning*: the car is driven manually around the track with a gamepad while the on-board camera frames and the corresponding driving actions are recorded; a convolutional neural network (CNN) is then trained to reproduce the mapping from image to action, so that the car can drive autonomously.

2. SYSTEM AND DATA

2.1. Hardware and track

The car is built around a Raspberry Pi with a camera and DC motors driven through an L298N motor driver, plus 3D-printed holders for the battery and screen (designed and stress-analysed in CAD/ANSYS). A DC-motor drive was chosen over a servo configuration for precise, unrestricted steering without the extra cost of a servo hat. The training track is a physical loop of roughly 1.7 m × 3.0 m with a 300 mm-wide white road over a dark background; a white road proved most robust to colour artefacts introduced by motion during recording.

2.2. Data collection and preprocessing

Training pairs of (camera frame, driving action) were recorded while driving the track manually with a gamepad, giving **1,495 training** and **374 validation** frames. Each frame is cropped and resized to 66×200 pixels (the PilotNet input size) and converted to an HSV/YUV colour representation, which is more robust to lighting and motion than raw RGB for this task.

3. MODEL

The starting point is the NVIDIA PilotNet CNN (five convolutional layers followed by fully-connected layers). PilotNet excels on large datasets but has drawbacks for this setting: it needs large amounts of labelled images (and, in the original work, multiple cameras), it can suffer from the "dying ReLU" problem, and its mean-squared-error (MSE) objective implicitly assumes Gaussian-distributed residuals — a reasonable assumption only when data is abundant. On the ≈1,900-frame dataset, the unmodified base model reached only 65%/57% train/validation accuracy.

Four targeted modifications were made (Fig. 1):

- **Steering as 3-class classification.** The single continuous-steering output was replaced by three discrete driving actions (left / straight / right). Because MSE is exactly the negative log-likelihood of a Gaussian, which fits a tiny dataset poorly, the objective was changed to a **categorical negative-log-likelihood (NLL)** loss over a softmax output — a more robust formulation for limited data (and the reason accuracy is the reported metric).
- **PReLU activation.** ReLU was replaced by the Parametric ReLU, whose learnable negative slope mitigates the dying-ReLU problem.
- **Lighter, wider-kernel convolutions.** The early convolutional filters were reduced (24/36/48 → 8/16/32) while the first kernel was widened (5×5 → 7×7) to capture a richer set of low-level features from few images, producing a lighter model that reduces overfitting on the small dataset.
- **Dropout** was kept between the dense layers as additional regularisation.

Fig. 1. *NVIDIA PilotNet (base) vs. the custom model. Amber-outlined blocks mark the changes: a wider first kernel, a lighter convolutional stack, a wider penultimate dense layer, and a 3-class classification head with a categorical NLL loss; ReLU is replaced by PReLU throughout.*

4. RESULTS

On the same self-collected dataset, the custom model substantially outperformed the unmodified base model, and the car completed the track reliably under autonomous control.

Table I. Accuracy on the self-collected dataset (1,495 train / 374 validation frames)

Model	Train acc.	Validation acc.
NVIDIA PilotNet (base)	65%	57%
Custom model (ours)	85%	81%

5. CONCLUSION

Adapting a large-data self-driving model to a small, self-collected dataset was the central challenge of this project. Reframing steering as a three-class classification with a categorical NLL loss, replacing ReLU with PReLU, and using a lighter convolutional stack with dropout raised validation accuracy from 57% to 81% and produced a car that drives the track autonomously. The main limitation is inherent to supervised behavioral cloning: the model does not generalise to unseen tracks, and the reported figures come from a single training run on a small validation set, so they should be read as indicative rather than statistically conclusive. Nonetheless, the project demonstrates an end-to-end pipeline — hardware assembly, data collection, computer-vision preprocessing, and model design — that learns to drive from little data.

Note: this is a condensed write-up of an undergraduate capstone project (2021), reconstructed for portfolio purposes; original code is no longer available.