Camera-based Line Following Robot using a Convolutional Neural Network

Steven Comandini
Electrical and Computer Engineering Department
Bradley University
Peoria, Illinois, USA
scomandini@mail.bradley.edu

***Abstract*—A camera-based line following robot uses only the images captured from a camera to detect and follow a lined path. In this paper, a system is designed and implemented on a robotic platform to be able to perform such a task. A convolutional neural network is trained and validated on manually collected data. It is then deployed onto a robotic platform which will interpret the predicted line class into motor movements. The system is shown to be successful with the model's validation accuracy at over 97% and the robot being able to follow the line.**

Keywords—Line Follower Robot, Microcontroller, Raspberry Pi, Webcam, CNN, Python, C, TensorFlow.

I. Introduction

There are many methods a robot could use to navigate through its environment. No matter what methodology or system is used, data is needed for the robot to understand its surrounding. A line following robot uses sensors to collect information about the ground in front of the robot in order to follow a distinctive line. In this paper, a webcam will be the only source of data a line following robot will receive to follow the lined path. Further explain in Sec. II, many techniques can be used when implementing a camera-based line follower. For this report, a convolutional neural network (CNN) will be implemented to interpret the images from the webcam. The CNN output is then sent to the robotics platform in order for the robot to move accordingly.

The paper is organized as followed. Sec. II discusses research already done using cameras on line following robots. Sec. III will lay out the process of creating and implementing the proposed system with Sec. IV containing the associated results. Sec. V concludes the paper and discusses further improvements.

II. Literature Review

Presented below are some published research on the topic of camera-based line following robotic platforms. They all use a camera as the input for the system but how they interpret that data is what differentiates them.

A. *Line Following Robot Using Image Processing*

J. Sarwade et al. in [1] proposed a solution to determine the correct actuation that a line follower robot has to take by the use of image processing. First, a photo of the line is captured. This is converted into a bitmap to identify the boundaries of the line. Next, a curve is calculated within the boundary. The slope of the curve is used to turn the robot and follow the line.

B. *Line Following Autonomous Driving Robot using Deep Learning*

In [2], R. Javanmard et al. describe a method of using a neural network to control an autonomous line following robot via camera. Data is first collected by having the robot drive around the track via remote control to collect photos and videos of the lines. These are then sorted into classes depending on the curve and position of the line relative to the camera. Next, a CNN and recurrent neural network (RNN) are trained based on the collected data. The accuracies of the CNN and RNN were compared. The system was then tested on the Webot simulator.

C. *Vision-based System for Line Following Mobile Robot*

A. H. Ismail et al. developed a system for line following robots using a camera along with image processing as seen in [3]. Unlike in [1], the lines from captured images are segmented and represented via a matrix. Using this matrix, a gain value is calculated. This gain is then used to determine the speed of each motor.

III. Implementation

The following subsections will explain the process of designing the neural network and deploying it on a robotic system.

A. *Software*

The CNN was written and trained via TensorFlow in Python 3.9.7 using the PyCharm IDE. The version of TensorFlow that was used was 2.8.0. On the robotic platform, Python 3.9.2 using the Thonny IDE was used in Raspberry Pi OS 11. The version of tflite used was 2.8.0. The Arduino IDE was also utilized on the robot for sending actuation to the DC motors.

B. *Data*

Since the images that the robot will capture is dependent on the physically platform itself, data to train the model will need to be manually collected. The following sections further describe the collection, preprocessing, and splitting process.

1) *Collection*

The robotic platform was manually driven around the track while taking photos of the line. OpenCV was used to capture and crop the image. Refer to Table I for the exact characteristics of the image.

TABLE I.
DATASET CHARACTERISTICS

Dimension:	Channel(s):	File Type:
256 x 128	1-CHANNEL (GREY SCALE)	JPEG

See Fig. 1 for the exact distribution of the collected dataset and Fig. 2 for example images.

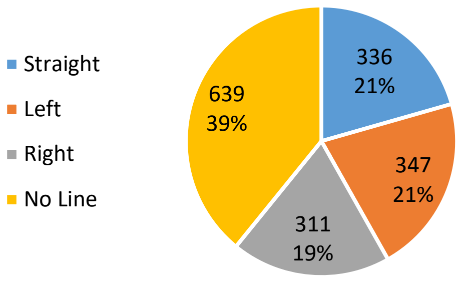


Fig. 1. Collected Dataset Class Distribution

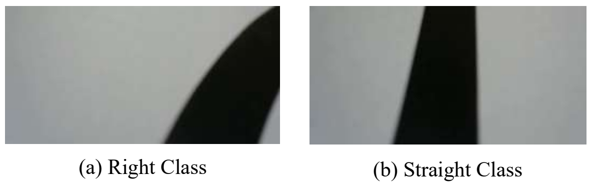


Fig. 2. Collected Images Example

2) Preprocessing

Before inputting the images to be trained in the TensorFlow model, preprocessing is applied to the images. The images were resized to 60x30 pixels using TensorFlow's resize function.

3) Splitting

The dataset was managed using TensorFlow's Data API. The data is split into 70% training, 20% validation, and 10% testing. The training batch size was 32 images.

C. Model

Since images will be the inputs for the neural network, a CNN was used for its ability to retain spatial information in an image. The architecture for the CNN can be seen in block diagram form in Fig. 3.

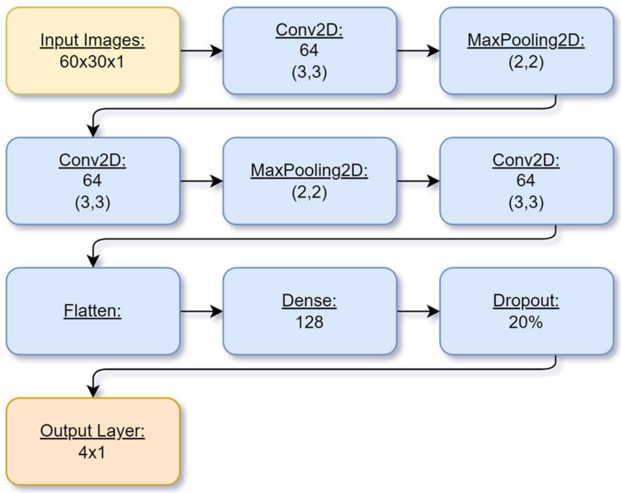


Fig. 3. Model Architecture

D. Hardware

Hardware will be used for training the model and deploying it. The following subsections will explain more about each case.

1) Training and Development

Data was collected using a Logitech C270 webcam. The model was trained on a Surface Book 2 Laptop. Its technical specifications include an Intel(R) Core(TM) i5-7300U processor and 8GB of RAM running on Windows 10 Pro.

2) Robotic Platform

The platform consisted of a Raspberry Pi 3B+ and the Logitech C270 webcam mounted on top of a SparkFun RedBot robotic chassis. The RedBot is powered by 4 AA batteries while the Raspberry Pi is powered via a 2S LiPo battery connected to a buck converter. See Fig. 4 for the completely assembled robot.

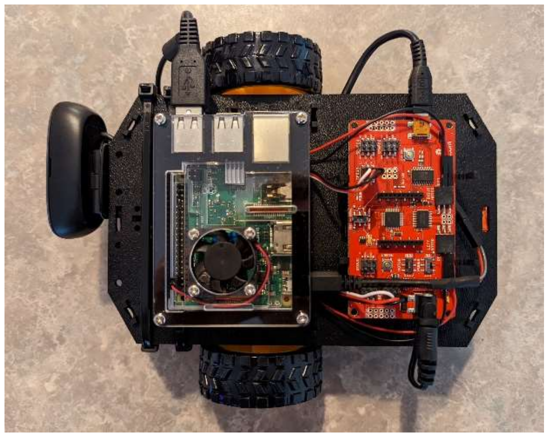


Fig. 4. Robotic Platform

E. Integration

After the model is trained on the laptop, it is saved as a tflite model and is transferred over to the Raspberry Pi. Fig. 5 showcases how the system works on the robotic platform. The red blocks are associated with the Raspberry Pi and the blue blocks are associated with the microcontroller on the RedBot. The Raspberry Pi is looping and capturing images at a rate of around 30 frames per second. While this is happening, the

RedBot is also looping in order to read serial data about the class of the line and how to move the motors accordingly.

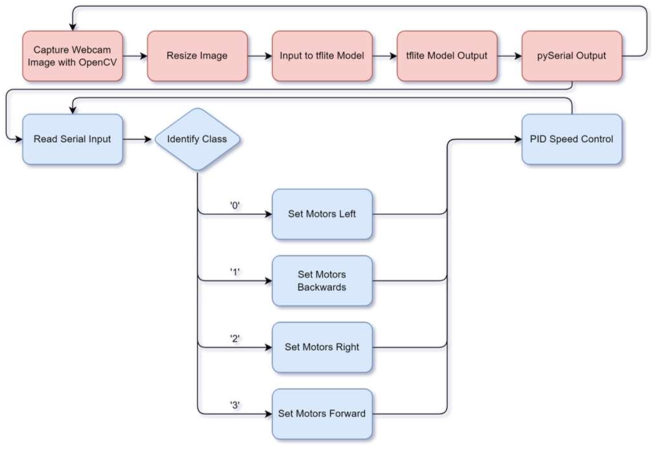


Fig. 5. Flow Chart of Robotic System

IV. Results

The subsequent subsections present results from training the CNN model and implementing it into the robot.

A. CNN Metrics

With the use of TensorFlow's early stopping, the model finished training after 26 epochs. As seen in Fig. 6, the model had a final training accuracy of 98.6% and a validation accuracy of 97.3%. Fig. 7 shows the model's loss over epochs as well.

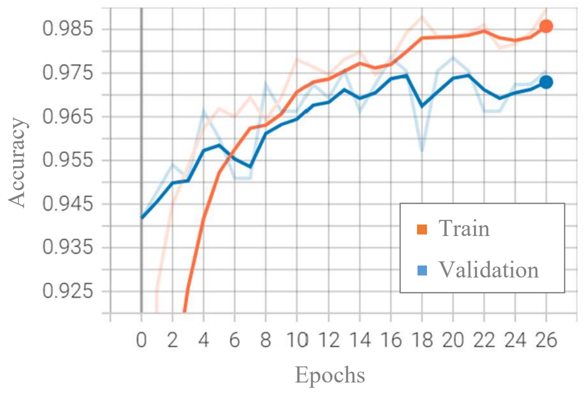


Fig. 6. Accuracy vs Epochs

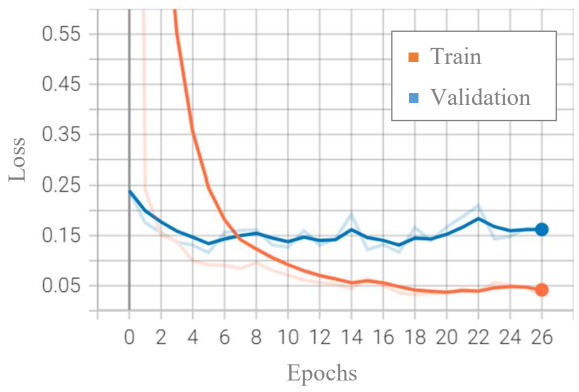
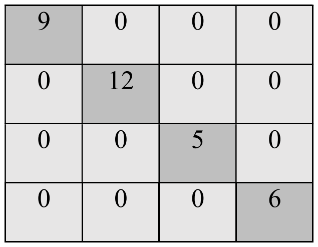


Fig. 7. Loss vs Epochs

After the model was trained, a batch of 32 images from the test dataset was tested against the model. Fig. 8 shows the confusion matrix for one batch size of images. Fig. 9 shows the output graphed alongside the batch of images. In this instance, the batch tested was predicted with 100% accuracy. This is not a surprise since the validation accuracy was over 97%.



9	0	0	0
0	12	0	0
0	0	5	0
0	0	0	6

Fig. 8. Confusion Matrix for 1 Batch of Images

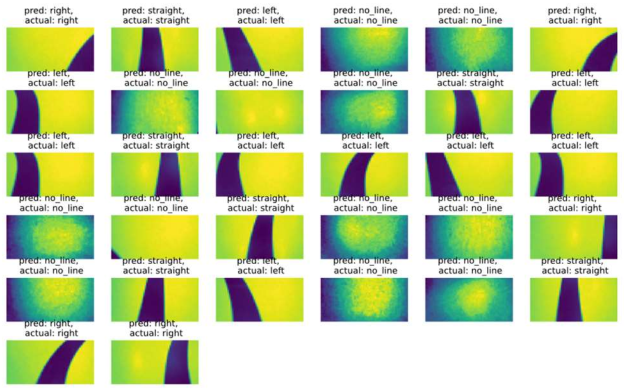

pred: right, actual: right
pred: straight, actual: straight
pred: left, actual: left
pred: no_line, actual: no_line
pred: no_line, actual: no_line
pred: right, actual: right
pred: left, actual: left
pred: no_line, actual: no_line
pred: no_line, actual: no_line
pred: no_line, actual: no_line
pred: straight, actual: straight
pred: left, actual: left
pred: left, actual: left
pred: straight, actual: straight
pred: left, actual: left
pred: left, actual: left
pred: left, actual: left
pred: left, actual: left
pred: no_line, actual: no_line
pred: no_line, actual: no_line
pred: straight, actual: straight
pred: no_line, actual: no_line
pred: no_line, actual: no_line
pred: right, actual: right
pred: no_line, actual: no_line
pred: straight, actual: straight
pred: left, actual: left
pred: no_line, actual: no_line
pred: no_line, actual: no_line
pred: straight, actual: straight
pred: right, actual: right
pred: right, actual: right

Fig. 9. Prediction Output Visualization for 1 Batch of Images

B. Robot Testing

The robot platform was able to follow the line, as seen in Fig. 10, using the logic defined in Fig. 5.

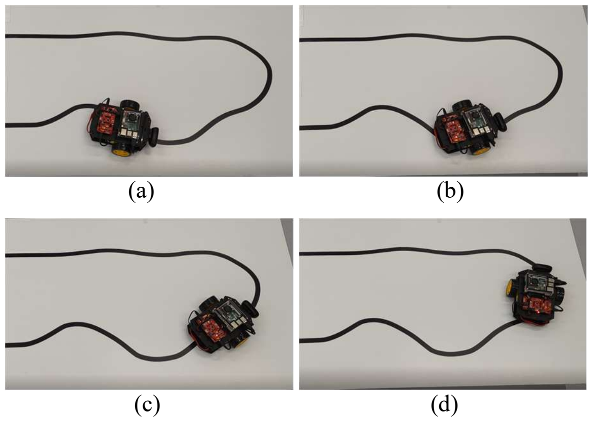


Fig. 10. Line Follower Robot Demonstration

V. Conclusion

A camera-based line following robot was successfully created using a convolutional neural network. Data was collected, processed, and inputted into the CNN. After the model was trained and validated, it was deployed onto the robotic platform.

There are improvements to the system that could be addressed in future iterations of the project. One issue that arose is when the robot lines up with a tight corner and the camera captures a nearly horizontal line. No data collected for training has horizontal lines so the model has to poorly guess the class for it. This results in the robot either going off course or switching between many classes and causing the robot to get stuck. Another improvement would be to combine both the Raspberry Pi and RedBot's microcontroller into one. If the system was running on one piece of hardware, it could perform faster since there will be no time delay in communicating between multiple devices.

References

[1] J. Sarwade, S. Shetty, A. Bhavsar, M. Mergu, and A. Talekar, "Line Following Robot Using Image Processing," in *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, Mar. 2019, pp. 1174–1179. doi: 10.1109/ICCMC.2019.8819826.

[2] R. Javanmard, A. H. Zabbah, M. Karimi, and K. Jeddisaravi, "Line Following Autonomous Driving Robot using Deep Learning," in *2020 6th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS)*, Dec. 2020, pp. 1–5. doi: 10.1109/ICSPIS51611.2020.9349547.

[3] A. H. Ismail, H. R. Ramli, M. H. Ahmad, and M. H. Marhaban, "Vision-based system for line following mobile robot," in *2009 IEEE Symposium on Industrial Electronics Applications*, Oct. 2009, vol. 2, pp. 642–645. doi: 10.1109/ISIEA.2009.5356366.