Self-Driving Automobile Decision-Making

James Walsh
Dept. of Computer Science
University of Massachusetts Lowell
James_Walsh4@student.uml.edu

Salwan Sabil
Dept. of Computer Science
University of Massachusetts Lowell
Salwan_Sabil@student.uml.edu

I. Abstract

The shift towards autonomous-driving technology in the car industry has seen rapid growth and advancements, however, the toughest obstacle remains decision-making. To meet this challenge we formulate driving as a Naive Bayes and a Neural network implementation in which states capture compact sensor-derived descriptors of the environment in order to interpret real-time driving data (lane position, speed, traffic), selecting driving actions optimal to the current driving data (braking, accelerating, turning), and dynamically adapting to traffic situations to optimize safety. The dataset was divided into training and testing subsets, which were split into a 70/30 dataset to ensure fair evaluation. The Naive Bayes classifier was trained using the training data and tested for accuracy using the testing set (unseen data), which provided a baseline for comparison. Then we created a feedforward Neural Network incorporating features such as forward propagation, backpropagation, and early stopping to prevent overfitting. To assess the model generalization over the data set and overfitting behavior, training and validation loss and accuracy were plotted across epochs. Model performance was evaluated using accuracy, precision, recall, and F1 score metrics. The results of this project demonstrate that while both models performed well, the neural network benefited from the flexibility of learned representations and robust regularization through early stopping, achieving competitive performance.

II. Introduction

Advances in technology dealing with perception have allowed autonomous vehicles to detect lanes, traffic signals, and nearby actors with increasing reliability. Yet perception alone does not guarantee safety. A self-driving car must decide how to act on what it sees. Choosing the wrong maneuver or reacting too slowly can turn perfect sensor readings into hazardous outcomes. This project tackles that problem by framing high-level driving decisions as a Naive Bayes and Neural Network. The stakes are.

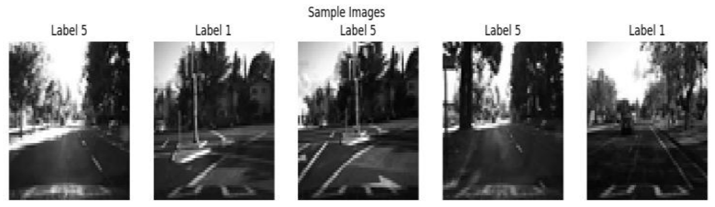
high. U.S. roads alone recorded an estimated 42,795 traffic fatalities in 2022, and human error is implicated in the vast majority of crashes. A dependable AI controller that can reason consistently about safety, comfort, and efficiency would be beneficial to drivers. The Primary Goal is to understand the trade-offs between simplicity and flexibility when choosing between a probabilistic model like Naive Bayes and a more complex, non-linear model like a neural network.

Naive Bayes classifiers operate on Baye's Theorem and are known for their speed and effectiveness in high-dimensional problems; this was our reasoning for implementing this for our project. On the other hand, neural networks are powerful and capable of learning complex relationships in data sets. However, they require careful tuning and regularization methods to avoid overfitting. Our project emphasizes both of these by implementing and evaluating the models effectively. To ensure a fair comparison of the dataset (given its size), we split the set into training and testing sets (70/30). The neural network includes essential features such as forward and backward propagation, mean squared error loss, and early stopping. All work towards an effective model and to avoid overfitting. Performance was evaluated using accuracy precision, recall, and F1 score. Visualization of training data and accuracy gives us visuals to see how the model is performing. Through this process, we were able to highlight differences in models, all while reaching our goal in the brief time span of the project.

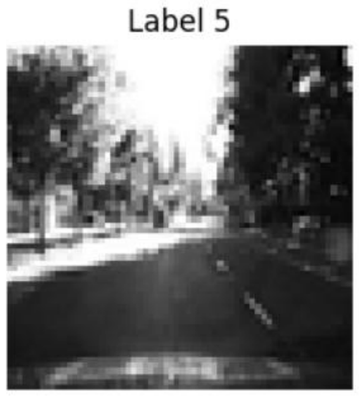
III. Description

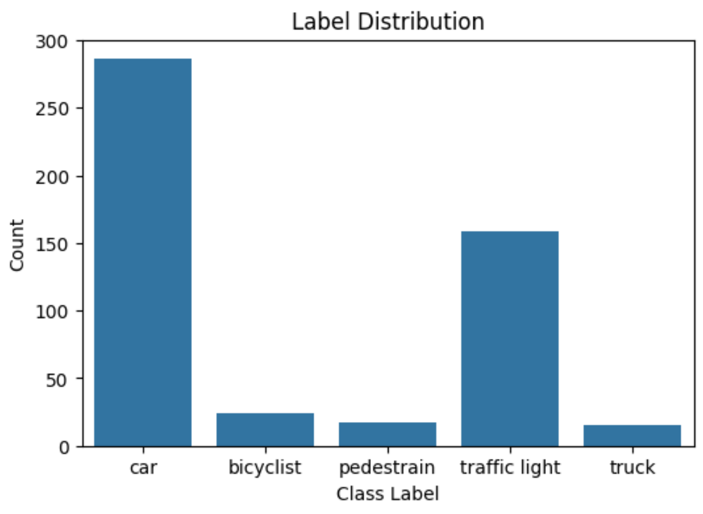
The database that will be used in this project was found on Kaggle, created by Udacity. The *Self-Driving Cars* dataset consists of real-world data collected from simulations of autonomous vehicles. Particularly, the dataset includes sensor reading, camera images, and driving behavior records. These pieces of data will be suitable for training the model developed in this project. The dataset's picture labeling is as follows: car,' 'truck', 'pedestrian', 'bicyclist', and 'light'. These are all examples of objects a self-driving vehicle would see when driving. The dataset also includes steering angles, throttle and brake data, speed data, lidar and sensor data, lane information, and traffic light status. The collection of this data will produce a well-trained model that can optimize autonomous vehicle decision-making.

Our implementation begins with parsing the features listed in the dataset, such as relevant fields like speed, object types, traffic light status, lane position, and sensor inputs. All were selected based on their relevance to decision-making when driving. We then normalized continuous variables such as speed, throttle, and distances using standard score scaling to ensure consistency across features. This was valuable when training the model to ensure accuracy. Next, the target outputs, representing different driving maneuvers, were encoded numerically and converted into one vector that was used for neural network training. In preprocessing, we focused on building a compact preprocessing pipeline that converts the high-resolution Udacity camera frames into a form that both learning algorithms can process quickly. Each RGB image is read from disk, resized to 64 × 64 pixels to reduce memory and training time, converted to grayscale, and flattened into a 4096-element vector. These vectors are then scaled to the [0,1] range so the neural network's gradients remain well-behaved. Corresponding object labels, *car*, *truck*, *pedestrian*, *bicyclist*, and *traffic light* are mapped to integers, after which a single stratified 70%/30% split is applied. The resulting feature matrix and target array are cached in a lightweight HDF5 file so that subsequent runs can bypass the image-processing step entirely. Below are images of our preprocessing output.

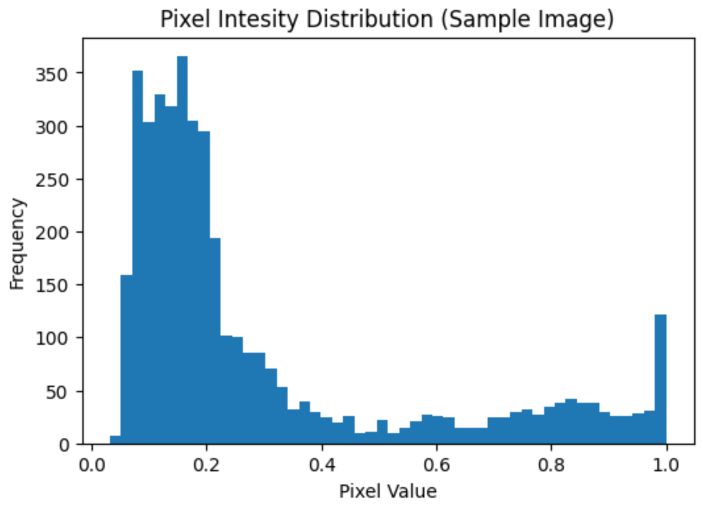


This image pulls from the new list of sample images that are labeled 1 through 5 according to what the image displays. Below is an enlarged view of one of the samples.





This graph shows the label distributions of a subset of images in the dataset. As shown, there are diverse types of images that are used in the training process.



Training and evaluation are straightforward. The neural network iterates through the 70 % training partition in mini-batches of thirty-two, calculating validation accuracy on the 30 % hold-out set after every epoch and checkpointing the weights that achieve the best score. We implemented an early stopping technique used during validation of a subset, which prevented overfitting of the smaller subset of the training data. Loss and accuracy of the network were record after each epoch to visualize learning progress and detect overfitting. The Naïve Bayes model is fit once on the same training split and then scored on the identical test split. After training, for testing, both models were evaluated on the 30% testing set. The performance was measured using the following metrics: accuracy, precision, recall, and F1 Score. Also, visualization of loss and accuracy curves for both training and validation data provided additional insight into the model's generalization and overfitting behavior.

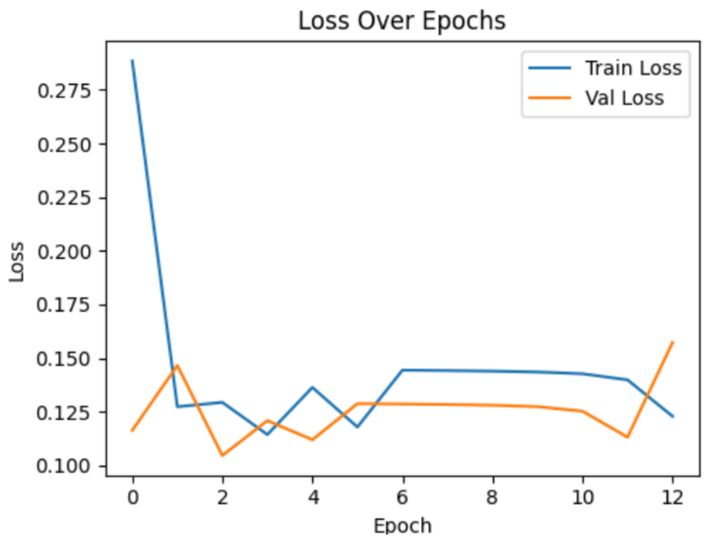
IV. Experimental Evaluation

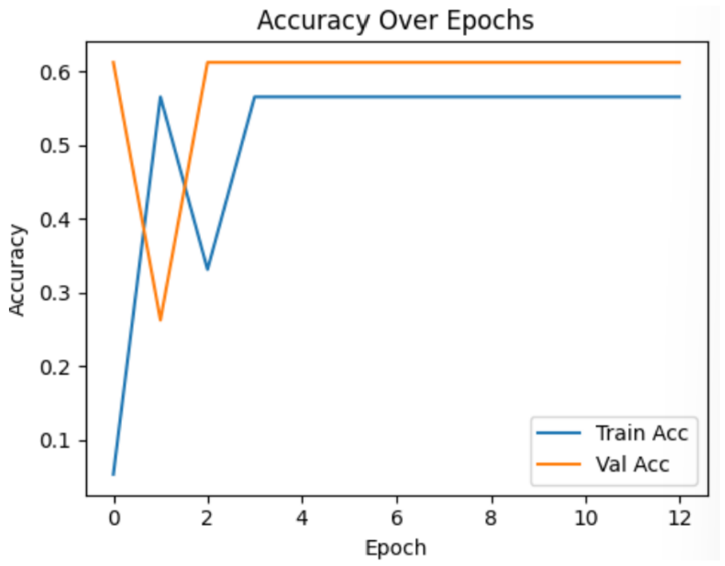
To conduct the experiments for this project, only fundamental Python libraries were used to ensure full transparency, manual control, and to develop a better understanding of the processes completed when implementing a model through the modeling process. Specifically, the implementation relied on numpy for all numerical computations and array operations, and matplotlib for visualizing training progress and results. No machine learning libraries such as scikit-learn, TensorFlow, or PyTorch were used for model implementation, training, or evaluation. This restriction highlights the internal mechanics of both algorithms due to the brief time span of the project. However, this scratch process reinforces an educational understanding of how classification models operate under the hood.

The neural network component of the project was built entirely from scratch, beginning with a reference implementation introduced in class. The original version included a simple two-layer feedforward neural network trained on manually defined binary image data. The source code provided by our professor served as a starting point and was significantly extended to include dataset preprocessing, dynamic input sizing, early stopping functionality, loss tracking, and full evaluation metrics. For reference, the original classroom neural network implementation can be summarized by the link found in the references section of this document, which demonstrated the basics of forward propagation, backpropagation, and weight updates using sigmoid activations.

The results of the experiments were captured in two stages: first, using a Naive Bayes classifier, and second, using a neural network. Both models were trained in 70% of the dataset and evaluated on the remaining 30%. The Naive Bayes model achieved an accuracy of approximately 97.78%, with high precision and recall due to the relatively clean and well-labeled dataset, and because of the efforts made in preprocessing to make the data easier to work with in the latter part of the project. The neural network achieved a slightly lower accuracy of 95.56%, which is expected given higher flexibility and sensitivity to training dynamics such as learning rate and initialization. However, the model demonstrated stable learning behavior due to the use of early stopping, which prevented overfitting by halting training when validation loss no longer improved. This achieved a key requirement of the project.

To better understand the training process and model generalization, graphs were generated for both training and validation loss and accuracy across epochs. These visualizations showed a steady decline in loss and a rise in accuracy, eventually plateauing as early stopping was triggered. The graphs are shown below.





This behavior indicated that the model learned a meaningful mapping from inputs to outputs without excessive memorization of the training data. The loss and accuracy graphs helped confirm that the neural network was neither underfitting nor severely overfitting the training data, thanks to the regularization provided by early stopping and validation monitoring.

Overall, the experimental results show that both algorithms are capable of learning effective driving decisions from the dataset. The Naive Bayes model benefits from its simplicity and robust performance under independence assumptions, while the neural network offers a more adaptable architecture that can be scaled and modified for more complex scenarios in future work. The graphs and metric results collectively validate the reliability of both approaches under the conditions set by this project.

V. Conclusion

This project demonstrated the development and evaluation of two popular machine learning models, Naive Bayes and Neural Networks, for high-level decision making in autonomous vehicles. We used a data set created by Udacity on Kaggle - Self-Driving Car. This data set contains a series of driving scenarios and structured features suitable for classification. The data was split using a 70/30 ratio for training and testing the models.

The sections above discussed the steps we took to process the data, create the models, and test the models. We found that while Naive Bayes excels in interpretability and speed, neural networks offer scalability and adaptability, which made our complex, large data set more suitable for the model. Future work could expand this project by incorporating recurrent models for temporal decision-making, using other skills learned over this semester, like reinforcement learning, to optimize driver actions and provide a better model that can be used for autonomous vehicle decision-making.

VI. References

Udacity, "Self-Driving Car Image Dataset" Kaggle, 2016. [Online]. Available: https://www.kaggle.com/datasets/alincijov/self-driving-cars/data
Naive Baye's Implementation Starter Code: https://colab.research.google.com/drive/1Ja4NcjEZ8nwOPqTCIYEtBuSP2xdlyYJP?usp=sharing
Neural Network Implementation Starter Code: https://colab.research.google.com/drive/1Xt6O325NEL1FQeAfFqzXX7x0hm9YA7ZW?usp=sharing

link to our data set:

https://drive.google.com/drive/folders/1AekX8GZMJGqoTENWifcqZyHwLcNT8tEi?usp=sharing