Machine Learning Homework 2: Car Racing

Karim Laiti

January 5, 2025

Contents

1 Introduction

The goal of this homework is to address a challenging machine learning task: controlling a racing car in a 2D simulated environment based on visual inputs. The project involves using image classification techniques to predict the appropriate driving actions, such as steering, accelerating, or braking, based on the input images. This task not only evaluates the capability of machine learning models in solving image classification problems but also their applicability to real-time decision-making scenarios.

The homework is structured in two main parts. First, the focus is on solving a 5-class image classification problem, leveraging convolutional neural networks (CNNs) for high accuracy and performance. Second, as an optional step, the learned model can be applied to drive the simulated car, further demonstrating its effectiveness in a practical environment.

This report outlines the methodology, experiments, and results obtained during the implementation of the project. It also includes an analysis of the hyper-parameter optimization process and the evaluation metrics used to measure the model's performance.

2 Dataset

The dataset used in this project is specifically designed for training and evaluating machine learning models in the context of autonomous car racing. It consists of labeled images corresponding to five driving actions, as detailed below. The images simulate a 2D racing environment, including road structures, turns, and other contextual elements relevant to decision-making.

2.1 Dataset Overview

- Total number of images: 9118 1.
- Image resolution: 96x96 pixels.
- Image format: PNG.
- Classes:
    1. Class 0: Do nothing.
    2. Class 1: Steer left.
    3. Class 2: Steer right.
    4. Class 3: Accelerate (Gas).
    5. Class 4: Brake.

Data Collection Process

The dataset was generated using a 2D racing simulation environment. Each image captures the perspective from the car's dashboard and includes visual cues such as the road,curves and turns. Each image is annotated with a corresponding action based on the ideal response to the visual input.

2.2 Data Splits

The dataset is divided into two subsets:

- **Training set:** Contains 6369 images, used for model training.
- **Validation set:** Contains 2749 images, used for evaluating the model's performance during development.

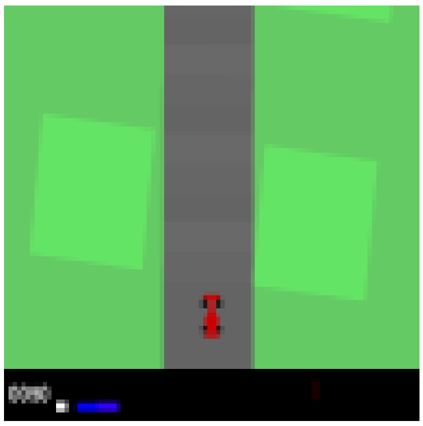

Figure 1: Sample images from the dataset.

2.3 Class Distribution

The class distribution highlights an imbalance, particularly for Class 4 (Brake), which is underrepresented in both the training and validation sets. This imbalance could impact model performance, especially for underrepresented actions.

2.4 Visualization and Analysis

To better understand the dataset, we analyzed the class distributions and visualized sample images. Figure 2 shows the histogram of the class distributions for the training and 3 validation sets. The visualization provides insights into the distribution of driving actions and highlights the imbalance in the dataset.

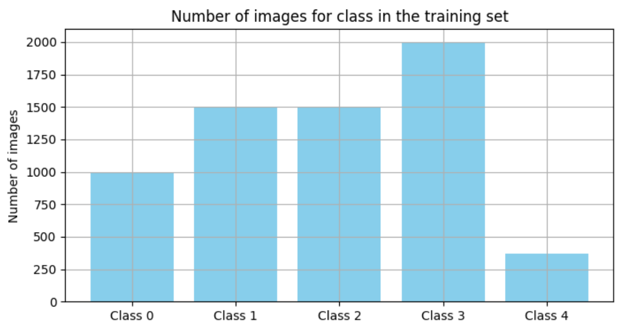

Figure 2: Class distribution in the training sets.

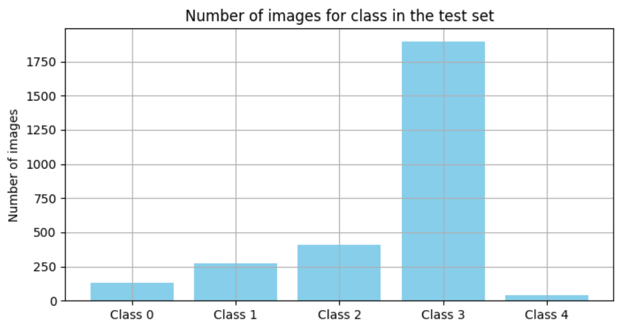

Figure 3: Class distribution in the validation sets.

2.5 Challenges and Considerations

The dataset presents several challenges:

- **Class Imbalance:** The underrepresentation of certain actions, especially Class 4, may lead to biased predictions. Addressing this could involve techniques like oversampling, data augmentation, or class weighting.
- **Noise in Images:** Images may include irrelevant elements (e.g., vegetation, dashboard) that could distract the model. Preprocessing steps, such as masking and grayscale conversion, aim to mitigate this issue.
- **Real-Time Applicability:** Since the model is intended for real-time decision-making, computational efficiency and latency must be considered during model selection and deployment.

2.6 Key Benefits of the Dataset

- Provides a realistic simulation environment for testing image-based decision-making models.
- Includes diverse scenarios, such as turns, straight paths, and obstacles, for robust learning.
- Offers a manageable dataset size, suitable for rapid prototyping and experimentation.

3 Preprocessing

Preprocessing is an essential phase to ensure that the model learns the relevant features from the images. The main steps applied during this phase are:

1. **Masking Green Pixels**: Green pixels, defined by a range of RGB values, are transformed to white (`[255, 255, 255]`). This helps remove distractions, such as vegetation or green signals, focusing attention on essential features.

2. **Uniforming Gray Pixels**: Gray pixels within a specific range are transformed to a uniform color (`[100, 170, 100]`), improving color consistency.

3. **Grayscale Conversion**: The image is converted to grayscale to reduce data dimensionality, then reconverted to RGB to maintain compatibility with deep learning models.

4. **Masking the Dashboard**: A rectangular bottom portion (height 12 pixels and width 96 pixels) is transformed to black (`[0, 0, 0]`), eliminating any irrelevant elements.

5. **Normalization**:

6. The pixel values are normalized to the range $[0, 1]$ to facilitate model training.

These steps improve the dataset quality by reducing noise, focusing learning on relevant features, and optimizing computational load.

3.1 Motivation and Benefits

- **Noise Reduction**: By neutralizing irrelevant colors and uniforming tones, the model better focuses on essential patterns.
- **Computational Efficiency**: Grayscale conversion reduces the amount of information per pixel, speeding up training and inference.
- **Focus**: By removing irrelevant environmental details, the model's ability to learn from structural features is improved.

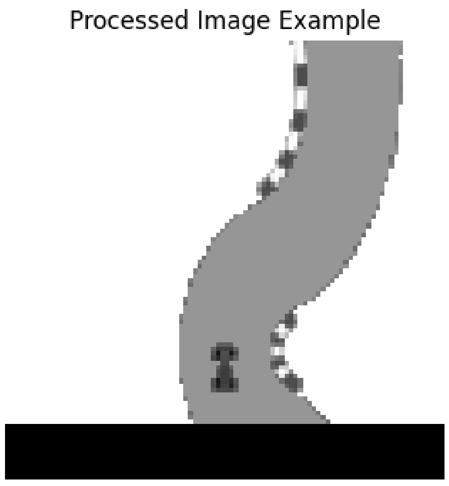

Figure 4: Illustration of the preprocessing steps applied to the images.

4 Model Description

The model used to classify the images in the dataset is a convolutional neural network (**CNN**) designed for image processing. The CNN consists of several convolutional, pooling, and fully connected layers to extract visual features and classify the correct action among the 5 possible ones.

4.1 Model Architecture

The model was implemented using the `TensorFlow/Keras` framework. Below are the details of the model structure:

- **Input Layer:**
  - Input size: $96 \times 96 \times 3$, corresponding to RGB images.
- **Convolutional Blocks:**
  - **First block:**
    - Convolutional layer with 16 filters of size 3×3, ReLU activation.
    - Followed by a max pooling layer (2×2) to reduce spatial dimensions.
  - **Second block:**
    - Convolutional layer with 32 filters of size 3×3, ReLU activation.
    - Max pooling layer (2×2).
  - **Third block:**
    - Convolutional layer with 64 filters of size 3×3, ReLU activation.
    - Max pooling layer (2×2).
  - **Fourth block:**
    - Convolutional layer with 128 filters of size 3×3, ReLU activation.
    - Max pooling layer (2×2).
- **Flattening Layer:**
  - Converts the three-dimensional representation of extracted features into a one-dimensional vector.

- **Fully Connected (Dense) Layers:**
  - **First layer:** 256 neurons with ReLU activation.
  - Includes a dropout layer with a probability of 0.5 to reduce overfitting.
  - **Second layer (Output Layer):** 5 neurons, one for each class, with softmax activation.

4.1.1 Hyperparameters and Considerations

- **ReLU Activation:** Used in convolutional and fully connected layers to introduce non-linearity.
- **Dropout:** Probability of 50% to prevent overfitting.
- **Optimizer:** An optimizer like Adam was used during training to minimize categorical cross-entropy.

4.1.2 Advantages of the Architecture

- Suitable for classifying low-resolution images due to hierarchical reduction of spatial dimensions.
- Balance between representation capacity (thanks to convolutional filters) and overfitting prevention (thanks to dropout).

5 Results and Analysis

In this section, we present the results of the model's performance on the dataset. We begin by showing the training and validation accuracy and loss curves, followed by a confusion matrix that provides further insight into the model's classification capabilities. Lastly, we include a classification report that summarizes the precision, recall, and F1-score for each class.

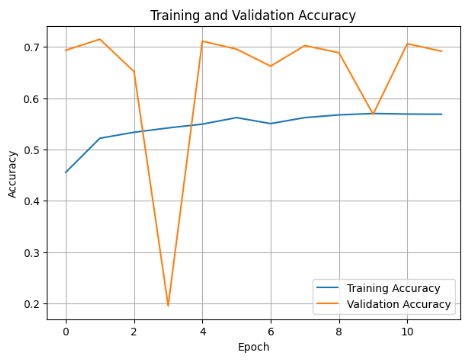

Figure 5: Training and validation accuracy

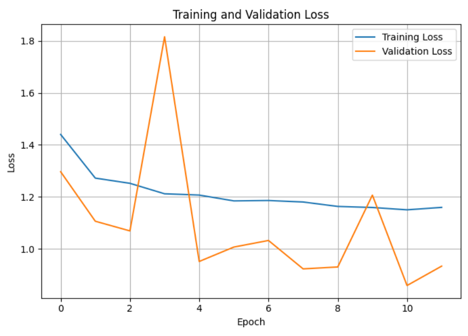

Figure 6: Training and validation loss

Figure 5 illustrates the accuracy curves for both the training and validation sets. We observe that the model achieves a steady increase in accuracy during training, with the validation accuracy also improving, indicating that the model is learning effectively without overfitting.

Figure 6 shows the loss curves for training and validation. The training loss decreases over time, and the validation loss stabilizes after a certain point, suggesting that the model is converging. A large gap between the training and validation loss could indicate overfitting, but in this case, the model's performance is balanced.

The confusion matrix in Figure 7 provides a detailed breakdown of the true and predicted classifications for the validation set. We notice that the model performs well for Class 3, with most instances correctly classified, while for Classes 0 and 4, there is a significant misclassification rate, which could be due to the relatively small number of samples in those classes.

Table 1 presents the precision, recall, and F1-score for each class in the dataset. Class 3 shows the highest performance, with precision and recall of 0.80 and 0.84, respectively, leading to a high F1-score of 0.82. On the other hand, Classes 0 and 4, which have fewer samples, suffer from poor performance, as evidenced by their low precision and recall values. This could be attributed to class imbalance, where the model may not have learned

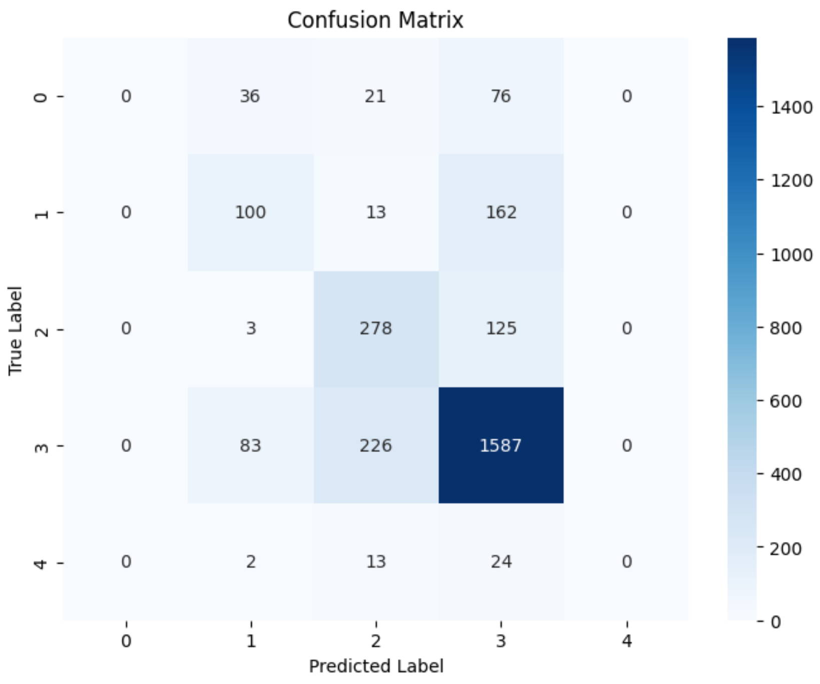

Figure 7: Confusion Matrix for the validation set

Table 1: Classification Report: Precision, Recall, F1-Score, and Support for Each Class

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.00	0.00	0.00	133
1	0.45	0.36	0.40	275
2	0.50	0.68	0.58	406
3	0.80	0.84	0.82	1896
4	0.00	0.00	0.00	39
Accuracy	0.71 (Overall)			
Macro Avg	0.35	0.38	0.36	2749
Weighted Avg	0.67	0.71	0.69	2749

to identify these classes effectively due to their lower representation in the dataset.

The overall accuracy of the model is 71, with a macro average F1-score of 0.36 and a weighted average F1-score of 0.69. These results suggest that the model performs reasonably well but could benefit from techniques such as data augmentation or class balancing to improve performance on the underrepresented classes.

In conclusion, the model demonstrates strong performance on Class 3, while its ability to classify minority classes (Classes 0 and 4) needs improvement. Further exploration into addressing class imbalance could lead to better overall results.

6 Conclusion and Future Work

In this project, a CNN-based approach was employed to classify driving actions from visual inputs, demonstrating promising results for the majority classes. However, class imbalance remains a challenge, emphasizing the need for techniques like oversampling or data augmentation. Future directions may involve extending the current approach to reinforcement learning, where agents learn optimal driving policies through continuous interaction with the environment, potentially leading to more robust decision-making and adaptability in complex, dynamic scenarios.