Introduction

This report describes the solution developed for the second homework assignment of the Machine Learning course. The task focuses on solving an image classification problem in a racing car environment using a Convolutional Neural Network (CNN).

The images represent scenes from a racing simulator, and the model must classify them into five possible actions:

- **0:** Do nothing
- **1:** Steer left
- **2:** Steer right
- **3:** Accelerate (gas)
- **4:** Brake

The dataset consists of RGB images (96x96) split across five classes, organized into folders by class labels.

The tasks performed for this homework are:

- Preprocessing and augmenting the dataset.
- Building a CNN model for classification.
- Evaluating the model using performance metrics.
- Discussing the results and possible improvements.

Data Preprocessing

Data Balancing

Initially, the dataset was unbalanced, with certain classes having more images than others. To address this, we randomly reduced the number of images in the overrepresented classes, ensuring an approximately uniform class distribution.

Data balancing is a critical preprocessing step in machine learning tasks, especially in classification problems where classes are imbalanced. In this racing car classification task, data balancing addresses several key challenges:

- Prevents Model Bias Toward Dominant Classes
- Improves Generalization
- Aids in Fair Metric Evaluation
- Enhances Stability During Training

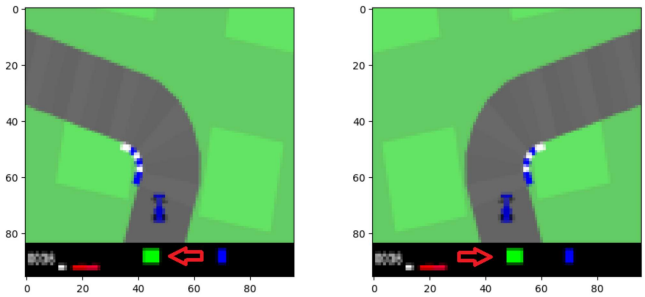
Data Augmentation

Horizontal flipping was selected as the most appropriate data augmentation strategy in order to enhance model training and broadening generalization of the model owing to the increase of the dataset size without altering the semantics of the actions performed by the vehicle.

A few classes warranted different approaches then:

- **classes 0 (do nothing), 3 (accelerate), and 4 (brake)**
  - These were added back into their respective original classes since flipping these images horizontally did not alter the meaning or context of the action. The model sees these actions as motionless or in other words rotation in a certain direction will have no directional preference hence augmenting increases the diversity of these datasets instead.
- **classes 1 (steer left) and 2 (steer right)**
  - These classes are affected by actions that are directional, hence Horizontal flipping of the images led to the opposite action. For this reason, the labels of these images were exchanged between classes about the other (for example steer left class was changed altogether to steer right).

Also, the green directional signal in the bottom panel of the image was flipped manually during augmentation. This ensured that after flipping, the visual cues of the action label remained consistent. For example, flipping an image from "steer left" to "steer right" included changing the signal of the panel to point to the right instead of the left.



Other augmentation techniques, like rotation or resizing, were avoided to maintain the spatial consistency between the car, the road, and the bottom panel, since these transformations may introduce inconsistencies in the visual cues. The focus was therefore on augmentations that retain the logical relationships within the image.

This augmentation strategy has balanced the dataset effectively, with maintenance of semantic and visual integrity of the data, improving model performance and better generalization.

Mark Edges

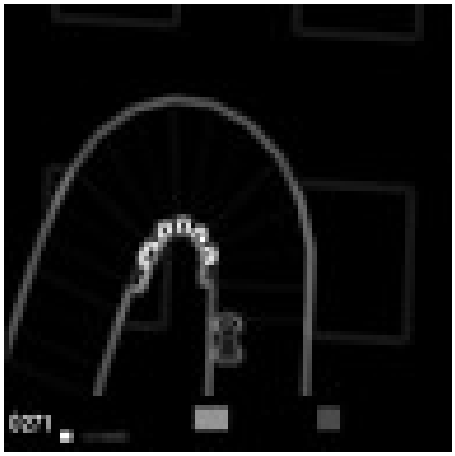
In addition to data augmentation, edge detection was applied to the dataset using the `Sobel` operator to preprocess the images and emphasize road features. Edge detection helps the model focus on structural information, such as lane boundaries and important contours, rather than unnecessary details like texture or lighting variations. Edge detection was applied only to the road area of the image, leaving the bottom panel (control indicators) unprocessed. This selective processing ensured that the focus was on road features, such as lane boundaries and curves, without affecting the possible crucial information in the control panel.

The steps for this approach were as follows:

1. **Image Splitting:** The image was split into two sections:
   - **Top Part (Road):** This section contains important features for predicting actions like steering and braking.
   - **Bottom Panel:** This part, which contains the model's control signals (e.g., directional indicators), was left unchanged.
2. **Apply `Sobel` Edge Detection:** The `Sobel` operator was applied to the top part of the image in both the horizontal and vertical directions to detect edges and gradients, with the following formula:

   $$G = \sqrt{G_x^2 + G_y^2}$$

   The resulting gradient map highlights structural details such as lanes and road boundaries.
3. **Recombine Processed and Unprocessed Parts:** The edge-detected top part was then recombined with the original bottom panel to create the final preprocessed image.



Advantages:

- **Focused Preprocessing:** Edge detection was applied only to the road, preserving the integrity of possible clues for the model in the control panel.
- **Enhanced Feature Extraction:** The model should more easily identify lane boundaries, turns, and other road features necessary for action classification.
- **Preservation of Panel Information:** The unchanged bottom panel allowed the model to rely on accurate directional and acceleration signals, avoiding confusion during training.

Notice: for the evaluation of the model trained with these images, also to the test set must be applied the marking of edges.

Final Datasets

To thoroughly evaluate the performance of the models, several versions of the dataset were created to explore how different preprocessing and augmentation strategies influenced the classification results. The datasets were balanced to ensure equal representation of all five action classes, and the following variations were tested:

Normal Data The original dataset consisted of RGB images (96x96 pixels) representing scenes from the racing simulation. Two variations of this dataset were created:

- **Normal Data (Non-Augmented):** The original images without any modifications.

Dataset Name	Description
Normal Data (Non-Aug)	Original RGB images without any augmentation.
Normal Data (Aug)	Augmented RGB images with class-specific flipping.
Marked Edges (Non-Aug)	Edge-detected images with the bottom panel intact.
Marked Edges (Aug)	Edge-detected images augmented with class-specific flipping.

- **Normal Data (Augmented):** The dataset was augmented using horizontal flipping. Images of actions unrelated to directional steering (*e.g., do nothing, accelerate, brake*) were flipped and retained within the same class. For directional actions (*steer left, steer right*), the flipped images were reassigned to their opposite class, with the corresponding adjustments made to the panel indicators to ensure logical consistency.

Marked Edges Data In this dataset variation, Sobel edge detection was applied to emphasize structural features like road edges and lane boundaries. The bottom panel was left unprocessed to preserve crucial control information. Two variations of the Marked Edges dataset were tested:

- **Marked Edges (Non-Augmented):** The images were processed with edge detection but left unaltered otherwise.
- **Marked Edges (Augmented):** Horizontal flipping was applied to the top part of the images (the road) in the same way as in the Normal Data augmentation process. The bottom panel remained unchanged, ensuring consistency in the directional signals.

Summary of Dataset Configurations By testing the models on these four dataset configurations, we sought to understand the impact of augmentation and edge detection on performance. This variety provided a comprehensive evaluation of the preprocessing techniques applied.

Models

This section details the architectures of the models tested for the classification task. The primary goal was to evaluate the effectiveness of different processing strategies and architectural enhancements in capturing relevant features from the dataset.

Single CNN Model

This model consisted of a **Convolutional Neural Network (CNN)** designed to process the entire image (both the road and the panel) without distinguishing between their roles.

Architecture:

- **Convolutional Layers:** The model had three convolutional layers, each followed by `ReLU` activation and `MaxPooling`:
  - Conv Layer 1: 8 filters, kernel size 3, stride 1.
  - Conv Layer 2: 16 filters, kernel size 3, stride 1.
  - Conv Layer 3: 32 filters, kernel size 3, stride 1.
- **Adaptive Average Pooling:** After the convolutions, an Adaptive Average Pooling layer reduced the spatial dimensions to 1×1, effectively extracting global features.
- **Fully Connected Layers:**
  - Hidden Layer: 32 neurons, `ReLU` activation.
  - Output Layer: 5 neurons, representing the 5 possible actions.

Reason for design: A simple architecture focuses on extracting task-relevant features, avoiding overfitting to noisy or unnecessary patterns. It provides a solid foundation for tasks where key features are easily identifiable, as in this project. Its computational efficiency, training stability, and interpretability further enhance its potential for generalization, making it a strong candidate for this task.

Dual CNN Model

The dual CNN model extends the Single CNN architecture by introducing two parallel CNN branches:

- One branch processes the **road (top part of the image)**.
- The other processes the **panel (bottom part of the image)**.

These branches were designed to focus on their respective roles:

- The **road branch** captured environmental features like lane markers and curves.
- The **panel branch** extracted control-relevant information like directional indicators.

Architecture:

- **Road Branch:** Identical to the Single CNN Model, with three convolutional layers (8, 16, 32 filters) and Adaptive Average Pooling.
- **Panel Branch:** Same architecture as the road branch but dedicated to the panel region.
- **Feature Fusion and Fully Connected Layers:**

- Outputs from both branches were flattened and concatenated.
- A shared fully connected network combined these features:
  * Hidden Layer: 128 neurons, `ReLU` activation.
  * Hidden Layer: 64 neurons, `ReLU` activation.
  * Output Layer: 5 neurons.

Reason for Design: This model leverages the specialized processing of distinct regions of the image. By splitting responsibilities, it potentially allows more accurate predictions for complex tasks that rely heavily on panel information.

Single CNN Model: Hyperparameter Search and Evaluations

The hyperparameter search was implemented to evaluate the performance of the **Single CNN** model. The goal was to identify the optimal combination of parameters that balances performance for the car racing task.

Four parameters were varied:

1. **Number of Epochs** ($[5, 10, 15, 20]$): The number of training epochs determines how long the model learns. Fewer epochs risk underfitting, while excessive training can lead to overfitting.
2. **Learning Rate** ($[0.01, 0.001, 0.0001]$): Controls the step size for weight updates during optimization. A moderate learning rate allows stable convergence without overshooting the optimal solution.
3. **Dropout Rate** ($[0, 0.25, 0.5]$): Dropout adds regularization by randomly dropping a proportion of neurons during training, preventing overfitting and improving generalization.
4. **Weight Decay (L2 Regularization)** ($[0.01, 0.001, 0.0001]$): encourage the model to keep the weights small, leading to a simpler model that may generalize better to new data.

Evaluation Metrics

To provide a comprehensive evaluation, the following metrics were used:

- **Accuracy**: The overall proportion of correctly classified samples.
- **Precision**: The ability of the model to correctly identify specific actions without producing false positives.
- **Recall (Sensitivity)**: The ability of the model to correctly identify all instances of specific actions.
- **F1-score**: The harmonic mean of precision and recall, balancing their trade-offs.

Results

The following table shows the best parameters chosen to maximize the trade-off between Accuracy and F1-Score.

Dataset Configuration	Epochs	Learning Rate	Dropout Rate	Weight Decay	**Accuracy**	**F1-Score**	**Precision**	**Recall**
Normal Data (Non-Aug)	20	0.01	0.25	$1e^{-4}$	46.95%	0.208	25.87%	29.27%
Normal Data (Aug)	20	0.01	0.5	$1e^{-4}$	48.51%	0.1	25.2%	20.19%
Marked Edges (Non-Aug)	20	0.01	0.25	$1e^{-4}$	49.12%	0.132	9.32%	24.12%
Marked Edges (Aug)	20	0.01	0	$1e^{-4}$	42.24%	0.13	21.57%	22.9%

Observations

The results from the hyperparameter search reveal several interesting insights:

1. **Effect of Epochs**:
    - Increasing the number of epochs beyond 15 does not significantly improve accuracy or F1-score across the datasets. This suggests that the model tends to converge earlier, and further training may lead to diminishing returns or overfitting.
    - Notably, **20 epochs** consistently yielded the highest performance compared to smaller values, balancing training time and model accuracy.

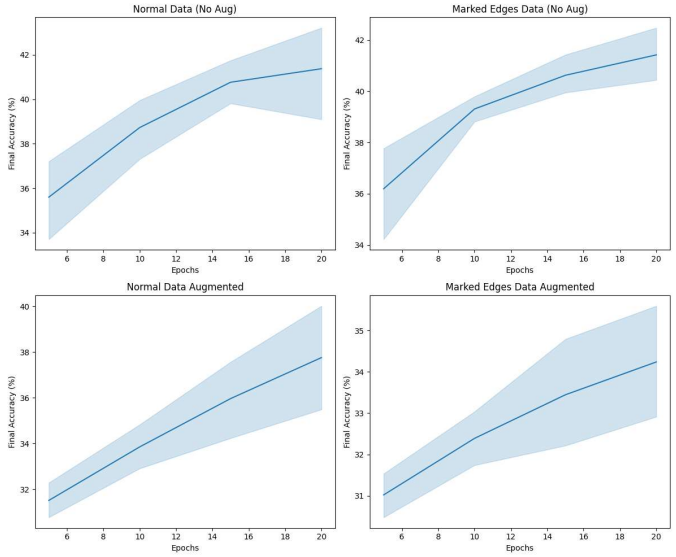


Figure 1: Accuracy trend in relation to the epochs (the blue shaded area represents the confidence interval, indicating the variability in the accuracy measurements)

2. **Impact of Dropout Rate**:

- A **dropout rate of 0.25** performed optimally for most datasets (both normal and marked edges), suggesting that a moderate regularization approach is beneficial for this task.
- The higher dropout rate of **0.5** decreased performance for **Augmented Data**, particularly impacting precision and recall, likely indicating that the model was too regularized to capture essential patterns.
- In contrast, a **dropout rate of 0** for the "Marked Edges (Augmented)" dataset resulted in lower generalization performance, with worse recall and precision scores, possibly indicating overfitting.

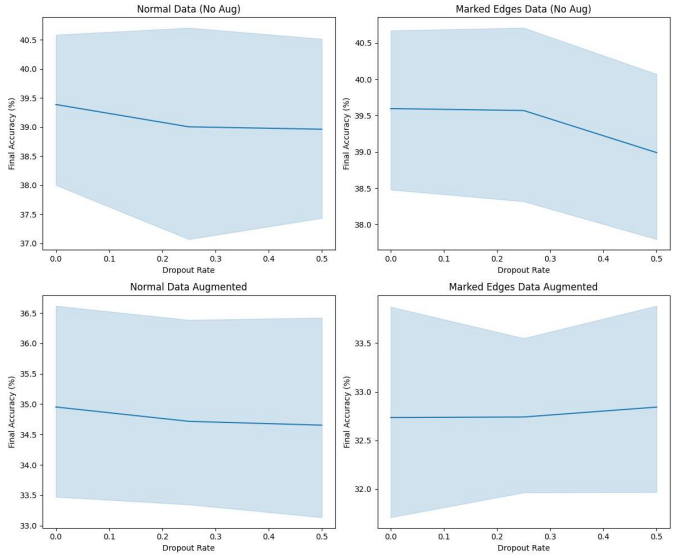


Figure 2: Accuracy trend in relation to the dropout rate

3. **Learning Rate Insights**: The learning rate of **0.01** generally produced the best outcomes across different datasets. However, lower values like **0.001** and **0.0001** led to a slower learning process without a noticeable improvement in generalization or model performance.
4. **Effect of Weight Decay**: Weight decay (L2 regularization) was held constant at a value of $1e^{-4}$ in most configurations, which seemed to provide a balance between model complexity and fitting capacity. No significant differences were observed by varying the weight decay, suggesting that the model might already be well-regularized with default parameters.
5. **Augmented vs. Non-Augmented Data**:
   - **Augmented Data** resulted in a noticeable drop in performance across the board (except in a few metrics), especially in terms of

Precision and **Recall**. This could indicate that data augmentation introduced noise or changed the distribution in a way that negatively impacted the model's ability to detect patterns effectively.

- For instance, the **Marked Edges (Aug)** dataset showed poorer performance in terms of **Precision** (25.2%) and **Recall** (20.19%). Further analysis is required to determine the cause of this drop – it could be due to either the augmented data introducing new, unseen conditions or the loss of important information during augmentation.

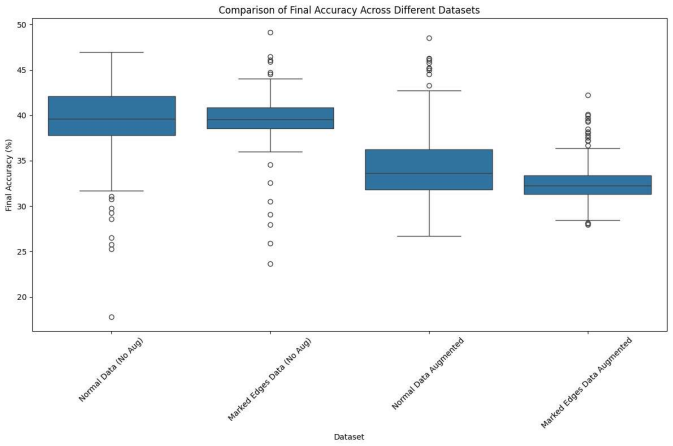


Figure 3: The blue shaded area indicates the confidence interval, reflecting the variability in the accuracy measurements with the white dots representing outliers that fall outside the typical range

Further Improvements and Next Steps

While the **Single CNN model** exhibited promising results, particularly with a learning rate of **0.01**, several avenues remain to refine the performance:

1. **Data Augmentation Exploration**: Future experiments could explore different augmentation strategies, particularly for more complex datasets like **Marked Edges (Aug)**, where performance seemed to degrade. Introducing new augmentation techniques or changing parameters (like augmentation intensity or patterns) could yield better results.

2. **Combining Precision and Recall**: For tasks like autonomous driving, balancing **Precision** and **Recall** is crucial. In the current configuration, models often favor **Accuracy** at the expense of low **Precision** or **Recall**. Future research may involve designing custom loss functions or weighting mechanisms that aim to find a more balanced trade-off between these metrics, perhaps using metrics like the **F1-score** or **Balanced Accuracy**.

Dual CNN Model: Hyperparameter Search and Evaluation

Since the Single model results, less parameters were varied for the **Dual CNN** model:

1. **Number of Epochs** ($[10, 20]$)
2. **Dropout Rate** ($[0, 0.25, 0.5]$): Dropout adds regularization by randomly dropping a proportion of neurons during training, preventing overfitting and improving generalization.

Results

Dataset Configuration	Epochs	Learning Rate	Dropout Rate	Weight Decay	**Accuracy**	**F1-Score**	**Precision**	**Recall**
Normal Data (Non-Aug)	20	0.01	0	$1e^{-4}$	40.65%	0.383	46.88%	44.17%

Observations

The results from the **Dual CNN** model highlight a few important findings compared to the **Single CNN** model. While the number of hyperparameters considered in the search was reduced, focusing on **epochs** and **dropout rate**, the **Dual CNN** architecture seems to have a positive impact on certain performance metrics.

- **Higher F1-Score**: The **F1-Score** improved to 0.383 for the **Normal Data (Non-Aug)** configuration with 20 epochs and no dropout. This indicates a better balance between **precision** and **recall** compared to the best results obtained in the **Single CNN** model. This improvement might be attributed to the dual approach, where two networks could potentially better capture the data's complexity and interdependencies, leading to a more nuanced classification output.
- **Accuracy**: Despite a higher F1-Score, the **accuracy** of **40.65%** is slightly lower than the **Single CNN** results from the **Normal Data (Aug)** configuration.
- **Effect of Dropout**: In comparison to the **Single CNN** model, where dropout was used in certain configurations, the **Dual CNN** model's results suggest that the lack of dropout improved precision, but resulted in reduced recall, highlighting the tradeoff between generalization and specificity.
- **Tradeoff Between Precision and Recall**: In the **Dual CNN**, the model has achieved a higher precision compared to the **Single CNN** model, but this comes at the cost of recall. This tradeoff suggests that while the model is more conservative in classifying positive instances, it may be missing some of the truly relevant samples.

Overall, the **Dual CNN** model shows interesting results, especially in terms of improving the balance between precision and recall.

Further Improvements

The next steps will involve experimenting with different dropout rates and optimizing the model's learning rate and weight decay to achieve better performance across all metrics. We will also consider incorporating data augmentation techniques to further enhance the model's generalizability, as it appears to be key for mitigating overfitting in similar architectures.

Conclusion

This project successfully implemented a CNN to classify images into five car racing actions. Preprocessing steps, including balancing and augmentation, were critical for achieving competitive performance. The proposed model demonstrates the effectiveness of deep learning for classification problems in simulation-based environments. Future work could focus on incorporating more advanced architectures and augmentation techniques for further improvement.

Future Work: Pretrained Models and Transfer Learning

As a future step, a plan is to evaluate the performance of **pretrained models** to explore if they can outperform this custom-designed CNN models. This will help to determine whether the used dataset contains sufficient patterns and features that a more complex model could leverage or if the dataset is too small, ambiguous, or noisy to provide meaningful signals for deep learning models. So this phase will provide insights into the following areas:

- Assessing whether **data augmentation** strategies could help improve performance.
- Investigating possible issues related to **dataset quality**, including inconsistencies, class imbalances, or limitations in diversity.
- Exploring the possibility that the architecture currently used might not be optimal for the self-driving task, and experimenting with other architectures may yield better results.

These steps will enable us to either confirm that our custom-designed model or the performed data augmentation is well-suited for the task at hand or gain valuable insights into the potential limitations of the dataset, allowing for further refinement in future experiments.