**Mohammed Faiq Shaikh -** #100905727 **Favour Eseagwu** - #100892897 **Bhaviya Chopra** - #100904417 **Gagandeep Singh -** #100897670

**Ashwin Philip** - #100906431

Self-Driving Car Game Project Report

# Introduction:

**AIDI-2005 Capstone Term II**

# MMP

The self-driving car project aims to develop an autonomous driving system using deep learning

techniques. The system utilizes a Convolutional Neural Network (CNN) based on the Nvidia model architecture. The CNN is trained to predict steering angles from input images captured by the car's cameras. The project involves data preprocessing, augmentation, model training, and evaluation. Through the development of an immersive gaming environment, the "Self-Driving Car Game with YOLO Object Detection" project hopes to emulate the capabilities of a self-driving car's perception system. The project improves player engagement and introduces a fresh gameplay dynamic by utilising the YOLO (You Only Look Once) object detection paradigm and integrating it into a gaming environment. The self-driving car's YOLO-based awareness allows it to recognise and respond to items in the gaming environment, making the experience more dynamic and realistic.

**Project objectives:**

* Use YOLO-based object detection to mimic the perception system of a self-driving car.
* Enable the self-driving automobile to communicate with and react to objects it detects to improve the gameplay.
* Annotate the game graphics with bounding boxes to see the results of the detection.
* Make a dynamic and engaging game about self-driving cars by utilising cutting-edge computer vision methods.

# Dataset:

The project uses a custom dataset collected from a simulated self-driving car environment. The dataset contains information about image paths and corresponding steering angles for the car's center, left, and right cameras. The data is stored in a CSV file with columns for "center," "left," "right," "steering,"

"throttle," "reverse," and "speed."

**Key features:**

* **YOLO Object Detection:**

Use the YOLO model from the ultralytics library to find items in the gaming environment using the YOLO object detection technique. Weights that have already been educated are used to ensure precise and effective item detection.

* **Gameplay Interaction:**

Allow the self-driving automobile to react dynamically to detected things and make judgements based on its perception of the objects and elements in its path.

* **Realistic Perception**:

By enabling the self-driving automobile to "see" and respond to its surroundings, you may give the game a realistic perception layer.

* **Visual Feedback:**

Drawing bounding boxes around detected items will give players visual input and help them better comprehend how a self-driving automobile sees the world.

**Technical Overview:**

* **Libraries and Dependencies:**

To handle image processing, visualisation, and display, the project depends on several libraries,

including NumPy, PIL (Python Imaging Library), ultralytics, and IPython.display.

* **Initialization of the YOLO Model:**

Pre-trained weights are used to initialise the YOLO model, providing object detection capabilities within the gaming environment.

* **Object Detection Workflow:**

The project detects objects in game photos, collects object data, and annotates bounding boxes around found items to visualise the findings.

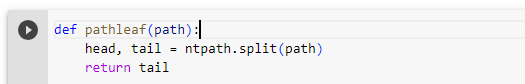
* **Interactive Gameplay:**

The interaction between the self-driving car and recognised items affects how it moves and makes decisions in the game world, which adds to the immersion and enjoyment of the game.

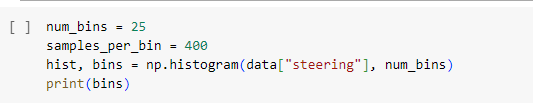
# Data Preprocessing:

Data preprocessing is crucial for training an effective self-driving car model. The following preprocessing steps are applied to the dataset:

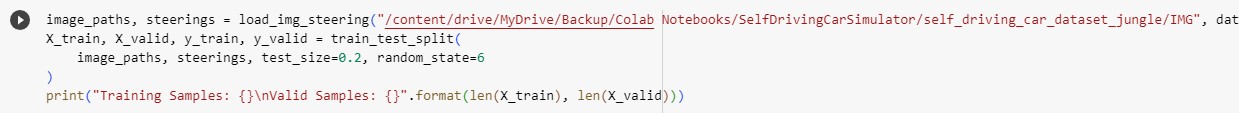
* Filename Extraction: Filenames are extracted from the file paths for easy manipulation.



* Steering Angle Binning: To address class imbalance, the steering angles are binned, and excess samples are removed from bins with a higher number of samples.



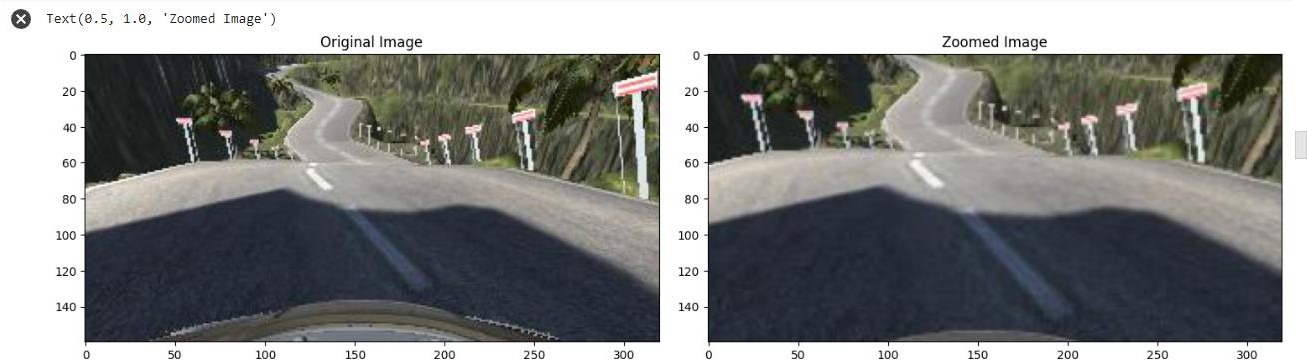
Train-Test Split: The dataset is split into training and validation sets to evaluate model performance.

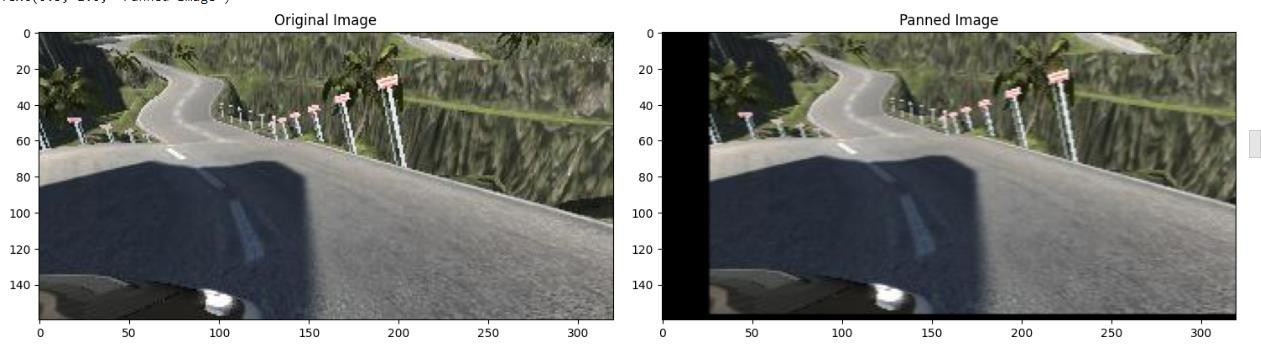


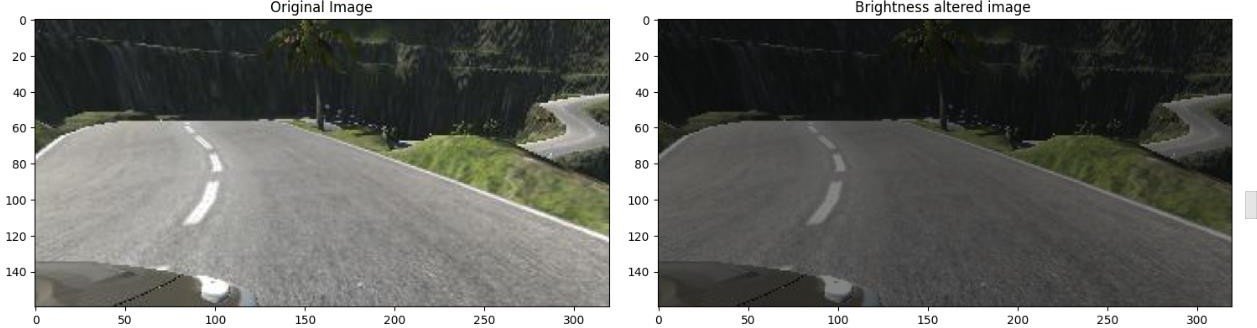
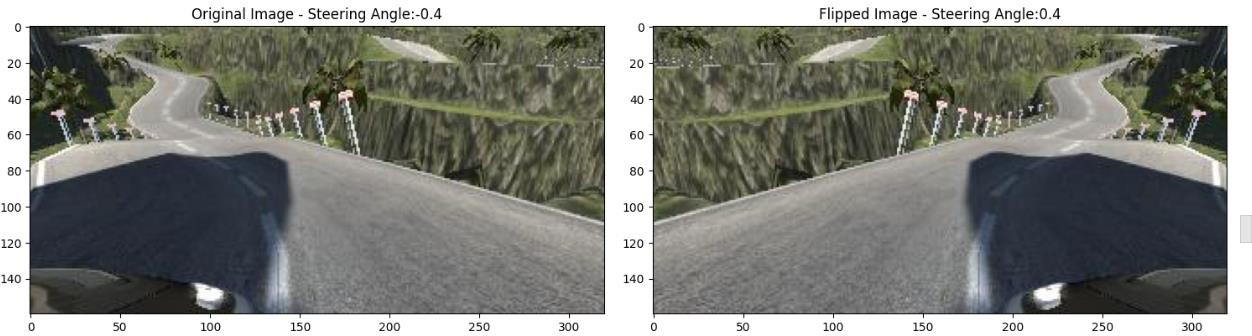
# Data Augmentation:

Data augmentation is employed to increase the diversity of the training data and improve the model's ability to generalize to various driving scenarios. Augmentation techniques include:

* Random Zooming: Randomly zooming into the image.
* Random Panning: Randomly translating the image horizontally and vertically.
* Random Brightness: Randomly adjusting the brightness of the image.
* Random Flipping: Randomly flipping the image and steering angle horizontally.



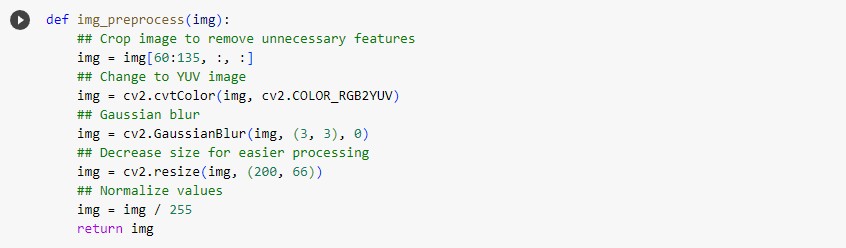




# Image Preprocessing:

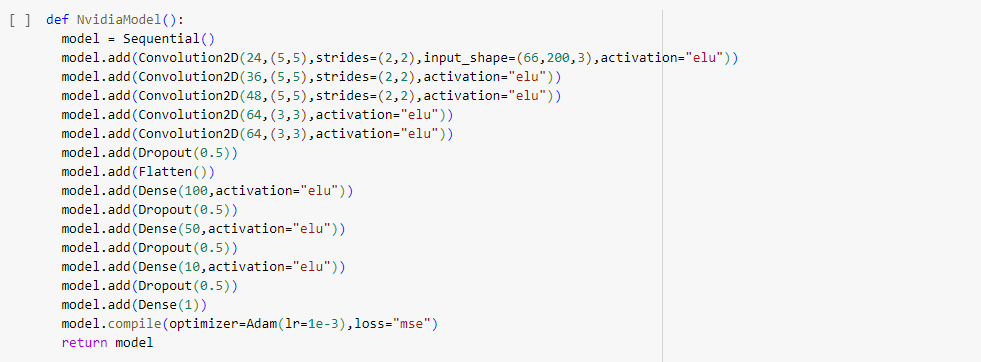
Images are preprocessed to focus on relevant features and reduce computational complexity. The following preprocessing steps are applied:

* Cropping: Unnecessary parts of the image are removed to focus on the road.
* Color Conversion: The images are converted to the YUV color space.
* Gaussian Blur: A Gaussian blur is applied to reduce noise.
* Resizing: The images are resized to a smaller resolution.
* Normalization: Pixel values are normalized to a range between 0 and 1.



# Model Architecture:

The CNN model is based on the Nvidia model architecture, which has been widely used for self-driving car projects. The architecture includes several Convolutional and Fully Connected layers with ELU activation functions to introduce non-linearity. Dropout layers are added to prevent overfitting. The model takes preprocessed images as input and predicts the steering angle as output.



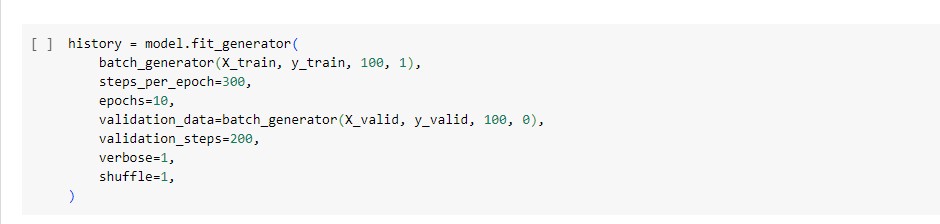
# Model Training:

The model is trained using the augmented and preprocessed images along with the corresponding

steering angles. The training is performed using a batch generator, which efficiently loads and processes images in batches to conserve memory. The model is compiled with the Mean Squared Error (MSE) loss function and the Adam optimizer. The training process is monitored using training and validation loss, and the model is saved after training.

# Evaluation and Results:

The trained model is evaluated on the validation set to assess its performance. The validation loss is analyzed to gauge how well the model generalizes to unseen data. If the model exhibits satisfactory performance, it can be deployed on a real self-driving car or a simulated environment.



# Screenshots:

# C:\Users\pc\Downloads\download (2).png

# C:\Users\pc\Downloads\download (3).png

# Conclusion:

The self-driving car project successfully develops an autonomous driving system using deep learning techniques. The implementation of the Nvidia model architecture, data preprocessing, augmentation, and training lead to a model capable of predicting steering angles from camera images. The project demonstrates the potential of using CNNs for self-driving car applications and lays the groundwork for future improvements and optimizations.

The "Self-Driving Car Game with YOLO Object Detection" offers a singular synthesis of advanced computer vision technology with engaging gameplay. The project raises the level of immersion, engagement, and enjoyment by mimicking a self-driving car's sensory system and incorporating it into the game experience. The idea has the potential to develop into an engaging and cutting-edge addition to the game industry with additional development and customisation.

# Future Enhancements:

Future enhancements to the self-driving car project may include:

* Collecting more diverse and real-world data to improve model generalization.
* Exploring additional data augmentation techniques for increased data variety.
* Fine-tuning hyperparameters and architecture to optimize model performance.
* Implementing advanced algorithms for lane detection, object recognition, and decision-making to enhance the overall self-driving system.