

Mohammed Faiq Shaikh - #100905727

Favour Eseagwu - #100892897

Bhaviya Chopra - #100904417

Gagandeep Singh - #100897670

Ashwin Philip - #100906431

AIDI-2005 Capstone Term II

MMP

Self-Driving Car Game Project Report

Introduction:

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.

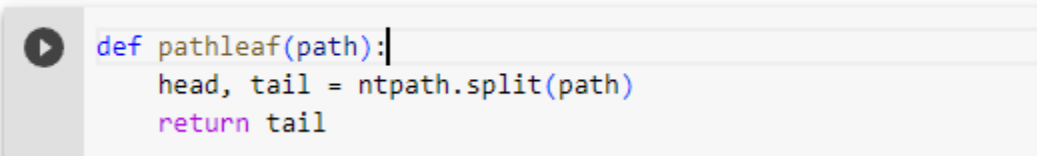
Dataset:

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."

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.



```
def pathleaf(path):  
    head, tail = ntpath.split(path)  
    return tail
```

- 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.

```
[ ] num_bins = 25
    samples_per_bin = 400
    hist, bins = np.histogram(data["steering"], num_bins)
    print(bins)
```

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

```
▶ image_paths, steerings = load_img_steering("/content/drive/MyDrive/Backup/Colab Notebooks/SelfDrivingCarSimulator/self_driving_car_dataset_jungle/IMG", dat
X_train, X_valid, y_train, y_valid = train_test_split(
    image_paths, steerings, test_size=0.2, random_state=6
)
print("Training Samples: {}\nValid Samples: {}".format(len(X_train), len(X_valid)))
```

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.



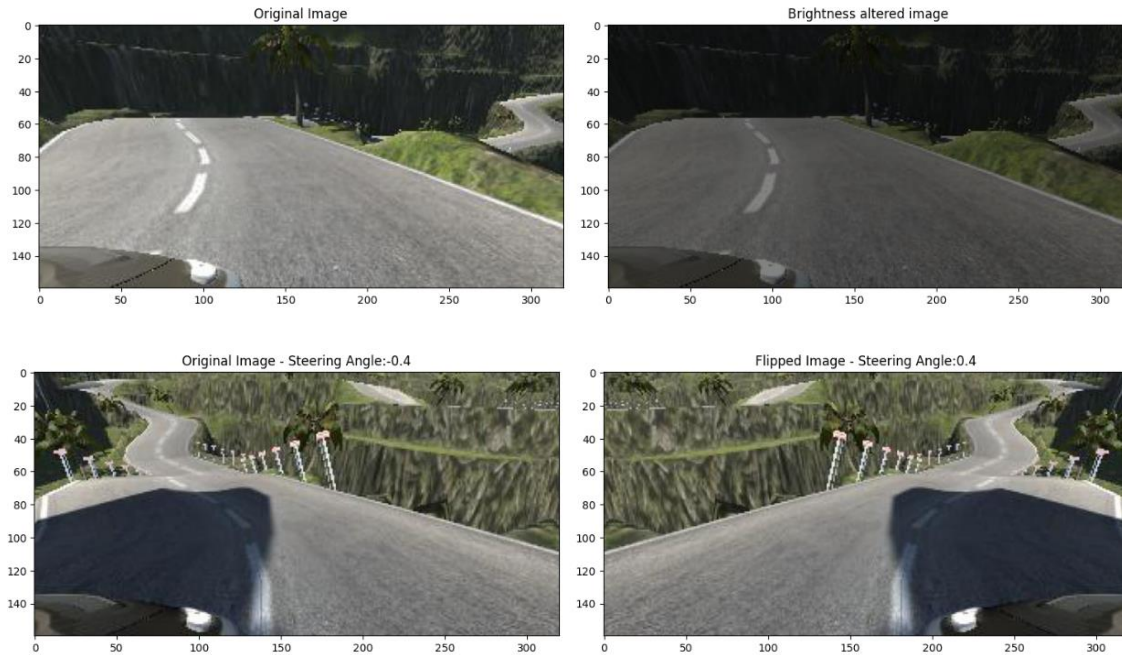


Image Preprocessing:

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.

```

def img_preprocess(img):
    ## Crop image to remove unnecessary features
    img = img[60:135, :, :]
    ## Change to YUV image
    img = cv2.cvtColor(img, cv2.COLOR_RGB2YUV)
    ## Gaussian blur
    img = cv2.GaussianBlur(img, (3, 3), 0)
    ## Decrease size for easier processing
    img = cv2.resize(img, (200, 66))
    ## Normalize values
    img = img / 255
    return img

```

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.

```
[ ] def NvidiaModel():
    model = Sequential()
    model.add(Convolution2D(24,(5,5),strides=(2,2),input_shape=(66,200,3),activation="elu"))
    model.add(Convolution2D(36,(5,5),strides=(2,2),activation="elu"))
    model.add(Convolution2D(48,(5,5),strides=(2,2),activation="elu"))
    model.add(Convolution2D(64,(3,3),activation="elu"))
    model.add(Convolution2D(64,(3,3),activation="elu"))
    model.add(Dropout(0.5))
    model.add(Flatten())
    model.add(Dense(100,activation="elu"))
    model.add(Dropout(0.5))
    model.add(Dense(50,activation="elu"))
    model.add(Dropout(0.5))
    model.add(Dense(10,activation="elu"))
    model.add(Dropout(0.5))
    model.add(Dense(1))
    model.compile(optimizer=Adam(lr=1e-3),loss="mse")
    return model
```

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.

```
[ ] history = model.fit_generator(
    batch_generator(X_train, y_train, 100, 1),
    steps_per_epoch=300,
    epochs=10,
    validation_data=batch_generator(X_valid, y_valid, 100, 0),
    validation_steps=200,
    verbose=1,
    shuffle=1,
)
```

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.

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.