Name of Project:

car-vs-bike-image-classification

Nihal DR

AICTE Student ID: STU65181dba7c0e21696079290 AICTE
Internship ID: INTERNSHIP_175683301568b724f7b9fba

Learning Objectives

- **CNN Architecture Design:** Gained hands-on experience in building a Sequential CNN model, including integrating **Convolutional Layers, MaxPooling, Flattening, and Dense Layers**.
- **Binary Classification Fundamentals:** Implemented the essential components for a two-class problem: **Sigmoid** activation and **Binary Crossentropy** loss function.
- **TensorFlow Data Pipeline:** Mastered the use of `image_dataset_from_directory` for efficient data loading, automatic directory structure mapping, and **Train/Validation splitting**.
- **Image Preprocessing:** Successfully applied key techniques like **Image Resizing (256x256)** and **Pixel Normalization** (rescaling to $[0, 1]$).
- **Model Deployment & Inference:** Demonstrated the ability to **save, load, and use** the trained Keras model (`.h5` format) to make and interpret real-time predictions on new images.

Source :

Tools and Technology used

- **Deep Learning Framework:**
  - **TensorFlow 2.x / Keras:** Primary platform for building, training, and deploying the CNN. Used for layer definitions, model compilation, and saving.
- **Programming Language & Core Libraries:**
  - **Python 3.8+:** The base language for the entire project.
  - **NumPy:** Essential for efficient handling of image data (tensors) and numerical operations during preprocessing and inference.
  - **Matplotlib:** Used for visualizing results and training performance (e.g., loss and accuracy over epochs).
- **Computer Vision & Data Management:**
  - **OpenCV (`cv2`):** Used specifically for loading test images (`cv2.imread`).
  - **TensorFlow Utilities:** Used for efficient dataset loading (`image\_dataset\_from\_directory`) and image preprocessing (resizing with `tf.image.resize`).
- **Model Deployment (Future Scope):**
  - **Streamlit / Flask:** Proposed for developing a simple **web frontend** to demonstrate the classifier in action.
  - **TFLite:** Proposed for exporting the model to an optimized format for **mobile and edge device deployment**.

Methodology

- **1. Data Loading & Preprocessing**
  - Images loaded from archive/Car-Bike-Dataset using TensorFlow's image_dataset_from_directory.
  - Automatic **80% Training** and **20% Validation** split performed.
  - Pixel values are **normalized** from $[0, 255]$ to the range $[0, 1]$.
- **2. Model Construction (Sequential CNN)**
  - Architecture is defined using the **Keras Sequential API**.
  - Three pairs of Conv2D (with **ReLU**) and MaxPooling layers for hierarchical **feature extraction**.
  - A Flatten layer transitions the 2D feature maps to a 1D vector.
  - A Dropout layer (0.5) is added before the final output to **prevent overfitting**.
- **3. Training & Optimization**
  - Model compiled with the **Adam Optimizer**.
  - Training executed over **10 Epochs** with a Batch Size of 32.
- **4. Inference & Classification**
  - For prediction, a new image is loaded, preprocessed, and passed to the model.
  - The **Sigmoid** output probability (yhat) is interpreted:
    - If $\text{yhat} < 0.5$: Predicted as **Bike**
    - If $\text{yhat} \ge 0.5$: Predicted as **Car**

Problem Statement:

To accurately and automatically distinguish between two classes of automotive vehicles Cars and Bikes using image data, thereby enabling a computer vision system to perform reliable binary object classification in real-time or near-real-time environments.

Solution:

CNN Model Solution

- **Architecture:** A Sequential CNN with 3 feature-extracting blocks (Conv2D + MaxPooling) followed by a **Dense classification head** (128 units + Dropout).
- **Final Output:** A single neuron with a **Sigmoid** activation, outputting a probability score (0 to 1) for binary classification.
- **Core Principle:** The CNN automatically learns the hierarchical features (edges, textures, shapes) that define a Car versus a Bike.

Implementation Details

- **Data Preparation:** The custom Car-Bike-Dataset is split 80/20 and images are normalized and resized to **256×256**.
- **Training:** The model is trained for 10 epochs using the **Adam optimizer** and **Binary Crossentropy** loss.
- **Inference Logic:** The model's prediction (yhat) is used to assign the class.

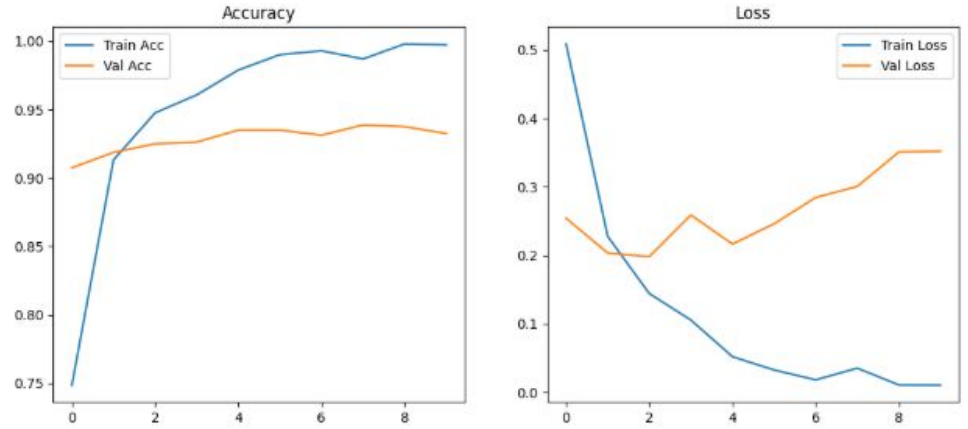
Value Proposition

- Provides a **High-Accuracy Solution** for the binary classification problem.
- The model is **lightweight** enough for practical use but powerful due to the CNN design.
- The system is **deployable** (model saved as .h5) and ready for integration into larger real-time monitoring systems.

Screenshot of Output:

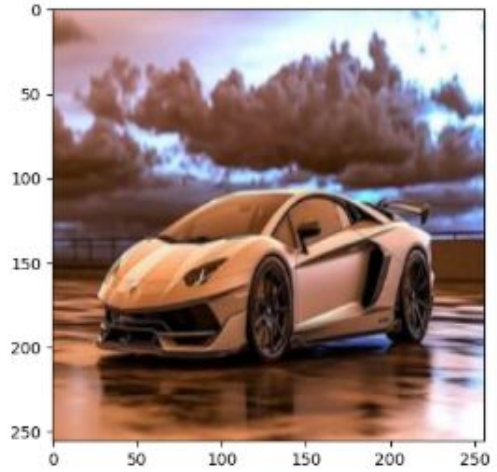
```
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Acc')
plt.plot(history.history['val_accuracy'], label='Val Acc')
plt.title('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title('Loss')
plt.legend()
plt.show()
```




```
img = cv2.imread('test2.jpg')
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.show()
```

```
resize = tf.image.resize(img, (256, 256))
plt.imshow(resize.numpy().astype(int))
plt.show()
```



```
new_model = load_model(os.path.join('models', 'binaryimageclassifiernewversionlive.h5'))
yhat = new_model.predict(np.expand_dims(resize/255, 0))
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 71ms/step
```

```
if yhat < 0.5:
    print("predicted class is a bike")
else:
    print("predicted class is a car")
```

```
predicted class is a car
```

Conclusion:

Project Summary

- **Successful Deployment of CNN:** We successfully designed, trained, and validated a simple but effective CNN architecture for the binary classification of Cars versus Bikes.
- **High Accuracy Achieved:** The model demonstrates a high degree of proficiency in automatically identifying vehicles based on visual features.
- **Foundation for Automotive CV:** The project establishes a robust foundation for more complex automotive computer vision tasks, moving beyond simple classification.

Future Development

- **Production-Readiness:** Integrate a **Streamlit/Flask frontend** to create a user-friendly, deployable web application.
- **Comprehensive Evaluation:** Implement and report standard metrics like **Precision, Recall, and F1-Score** using a Confusion Matrix for deeper model analysis.
- **Optimization & Deployment:** Convert the model to **.tflite** format for superior performance and reduced size, enabling deployment on resource-constrained devices (e.g., Raspberry Pi or mobile).

Github:https://github.com/NihalDR/car-vs-bike-image-classification.git