Car Type Classification

AI Skills Project Report

December 23, 2025

Abstract

This report documents the design, implementation, and deployment of a Deep Learning system for fine-grained car classification. Using Transfer Learning with EfficientNetB4, the team achieved a validation accuracy of 86.3% and robust performance on the test set. The project includes a fully functional Streamlit GUI with real-time inference and Grad-CAM explainability.

Contents

1 Project Overview

1.1 Executive Summary

This project aims to develop a Deep Learning system capable of recognizing car makes, models, and years from images. Unlike generic object classification, car type recognition is a **fine-grained classification** problem, where classes (e.g., *2012 BMW 3 Series* vs. *2012 BMW 5 Series*) share highly similar visual features.

Our team successfully designed, trained, and deployed a solution using **Transfer Learning** with state-of-the-art Convolutional Neural Networks (CNNs). The final deliverable includes trained models, a comparative performance analysis, and a fully functional Graphical User Interface (GUI) built with **Streamlit**.

1.2 Problem Statement

- **Goal:** Classify 196 distinct classes of cars from the Stanford Cars Dataset.
- **Challenges:** High inter-class similarity, varying lighting conditions, and diverse camera angles.
- **Applications:** Automated parking management, traffic surveillance, and automotive inventory systems.

2 Team Structure

Team Member Name	ID
Youssef Atef Tayh	931230366
Omar Yasser Sayed	932230124
Ismail Ibrahim Ismail	931230063
Ahmed Ali Saber	931230039
Ahmed Ibrahim Fahmy	931230014

Table 1: Project Team Members

3 Methodology

3.1 Dataset Preparation & Splitting

- **Source:** Stanford Cars Dataset
- **Total Images:** 16,185 images divided into 196 classes.

To ensure robust evaluation, the dataset was explicitly divided into three subsets:

- **Training Set:** Used for learning model weights (approx. 80% of training data).
- **Validation Set:** Used for hyperparameter tuning, monitoring overfitting, and early stopping during training (approx. 20% of training data).

- **Test Set (8,041 images):** Completely unseen data used **only** for final evaluation metrics after training is complete.

Preprocessing & Augmentation:

- Images were resized according to the specific input requirements of each architecture.
- Pixel values were normalized to the range [-1, 1] or [0, 1].
- **Data Augmentation:** Rotation (15°), Zoom (10%), and Horizontal Flip were applied to prevent overfitting.

3.2 Training Configuration (Hyperparameters)

We implemented a consistent training pipeline across all architectures. The specific hyperparameters for each model are detailed below:

Model	Best Epoch	Batch Size	Optimizer	Learning Rate
InceptionV3	19	16	Adam	$1e^{-5}$
ResNet50	20	16	Adam	$1e^{-5}$
EfficientNetB4	20	16	Adam	$1e^{-4} \to 1e^{-5}$

Table 2: Hyperparameters and Training Details

4 Experimental Results

This section presents the detailed performance metrics for the three experimented architectures.

4.1 InceptionV3

- **Input Size:** 299x299
- **Training Accuracy:** $\approx$ 99%
- **Best Validation Accuracy: 86.9%**
- **Observations:** Achieved high accuracy with limited overfitting. Fine-tuning was enabled.

4.2 ResNet50

- **Input Size:** 224x224
- **Training Accuracy:** $\approx$ 99%
- **Best Validation Accuracy:** 80.9%
- **Observations:** Good baseline performance, but limited by the smaller input resolution (224x224) compared to other models.

4.3 EfficientNetB4 (Best Model)

- **Input Size:** 384x384
- **Training Accuracy:** 98–99%
- **Best Validation Accuracy:** 86.3%
- **Techniques Used:** Multi-stage training, Label Smoothing (0.1).
- **Observations:** Provided the best balance between accuracy and fine-grained feature extraction due to higher resolution and compound scaling.

Model	Input Size	Val Accuracy	Training Acc
InceptionV3	299x299	86.9%	$\approx$ 99%
ResNet50	224x224	80.9%	$\approx$ 99%
EfficientNetB4	384x384	86.3%	98–99%

Table 3: Summary of Model Performance

5 Graphical User Interface (GUI) & Explainability

We developed a professional GUI using **Streamlit** to demonstrate the model's capabilities in real-world scenarios.

5.1 Grad-CAM Explainability

To ensure the model's reliability, we integrated **Grad-CAM (Gradient-weighted Class Activation Mapping)**.

- **Why Grad-CAM?** It provides transparency by visualizing ”where” the model is looking inside an image.
- **How it works:** It uses the gradients of the target class flowing into the final convolutional layer to produce a coarse localization map.
- **Benefit:** It helps verifying that the model is learning relevant features (e.g., headlights, grill, body shape) rather than relying on background noise or irrelevant patterns, thereby increasing trust in the predictions.

5.2 Interface Features

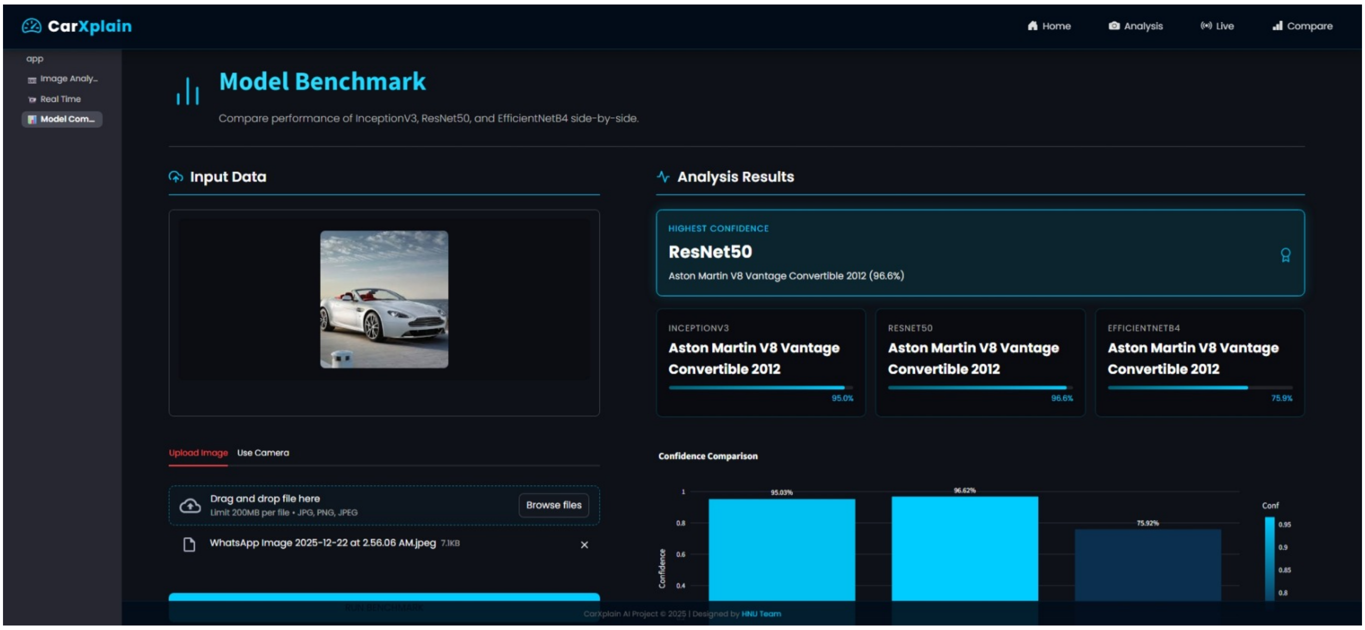

Figure 1: Main GUI Interface
(Main interface showing image upload and Top-3 probabilities)

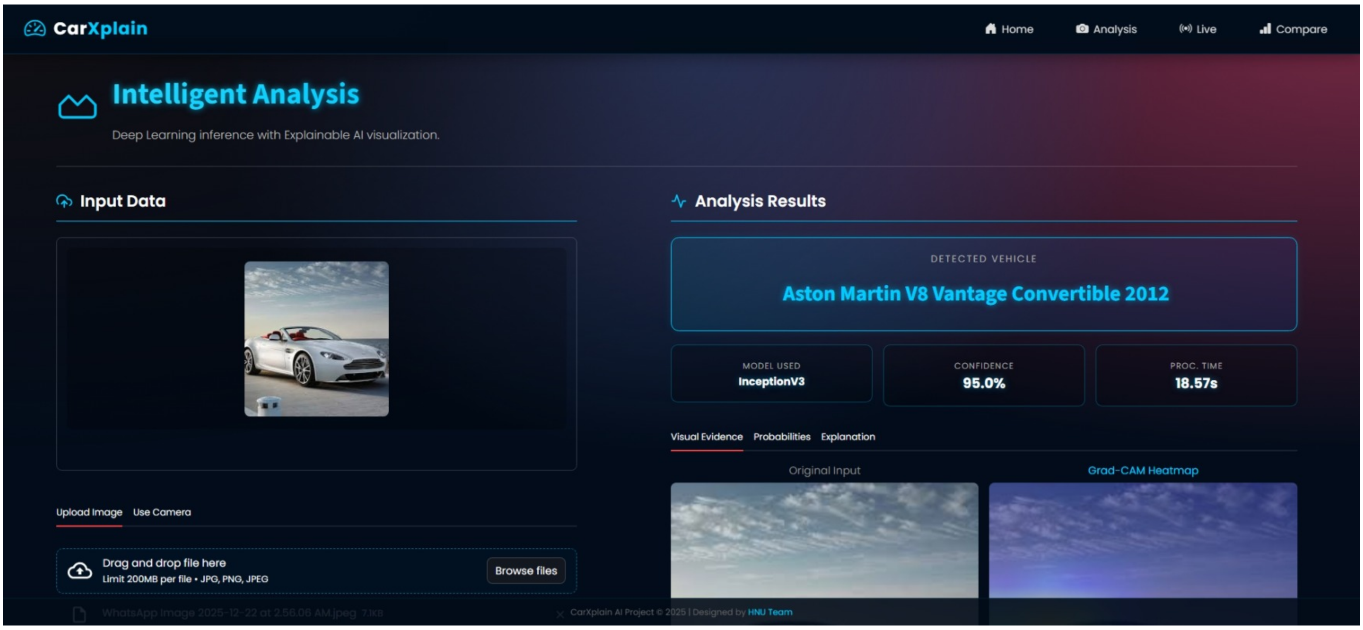

Figure 2: Grad-CAM Visualization
(Heatmap visualization highlighting car features)

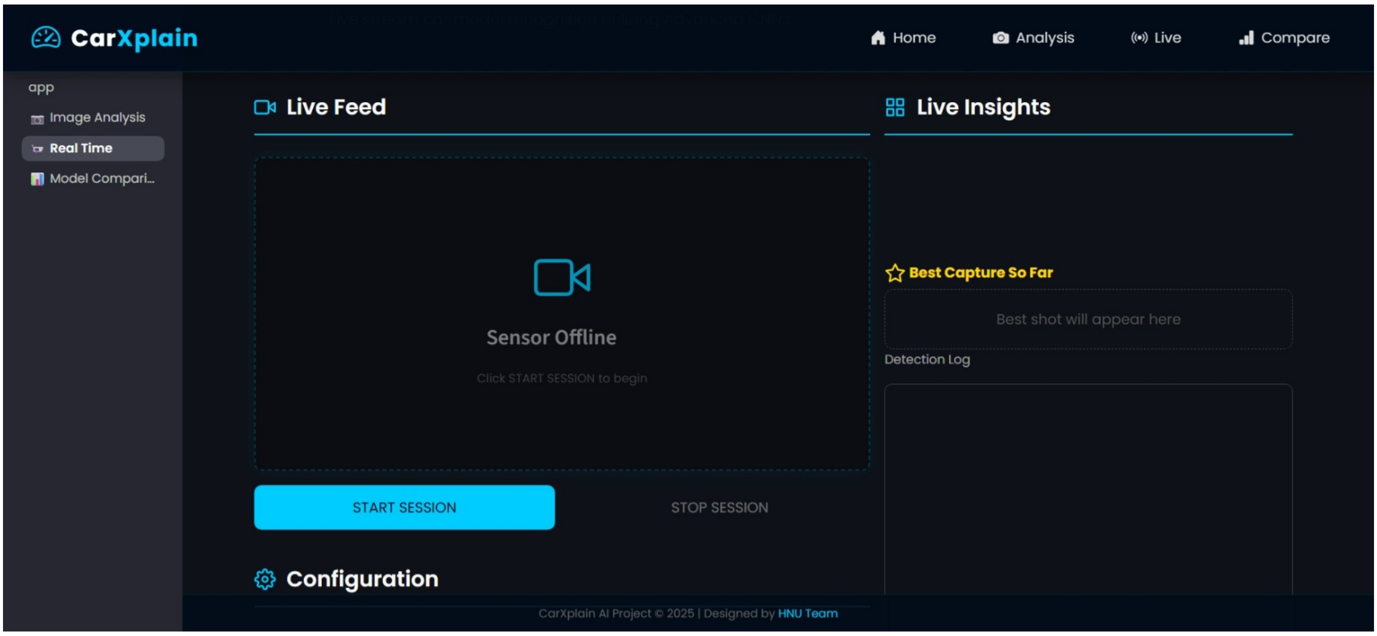

Figure 3: Real-Time Webcam Inference

6 Challenges & Solutions

- **Challenge 1: Overfitting.**
  - *Solution:* Implemented heavy Data Augmentation and Label Smoothing (0.1) in EfficientNet.
- **Challenge 2: Visual Similarity.**
  - *Solution:* Used high-resolution inputs (384x384) to capture minute details.
- **Challenge 3: Real-time Performance.**
  - *Solution:* Optimized model loading in Streamlit using caching (`@st.cache_resource`).

7 GitHub Repository

The complete source code, datasets, and models are organized in our GitHub repository: CarXplain - AI Skills Project.

8 Conclusion

The **Car Type Classification** project successfully demonstrates the power of Transfer Learning. By comparing **InceptionV3 (86.9%)**, **EfficientNetB4 (86.3%)**, and **ResNet50 (80.9%)**, we deployed a robust solution capable of fine-grained recognition, supported by an explainable AI interface.