DETACTX

GRADUATION PROJECT

DEPI ROUND 3 MICROSOFT MACHINE LEARNING TRACK

Submitted By:

- Basel Mohamed Mostafa
- Omar Yasser Sayed
- Zeyad Gamal Mohamed
- Abdelrahman Kamal Elkhabery
- Mohamed Hamada Farghali
- Ziad Ahmed Samir

Abstract:

- Car Classification & Object Recognition Website.

Internship Program:

Microsoft Machine Learning Engineer

11/30/2025

CONTENTS

CHAPTER 1: PROJECT ABSTRACT

With the rapid advancements in computer vision and deep learning, automated image understanding has become a core component of modern intelligent systems. This project presents the **DetactaX Vision App**, a web-based application that integrates **fine-grained car classification** and **real-time object detection** using state-of-the-art deep learning models.

This app utilizes two main classification Neural Networks along with the YOLO12X for Object Detection, the two main models are:

1. **MobileNetV2**: For Baseline Testing.
2. **ResNet50**: Main Car classifier.

We utilize a special and unique method where our Baseline Testing model is trained on **CIFAR-10** Dataset as a Control/Test Classifier and our ResNet50 model is trained on the renowned **CUHK-CompCars** dataset to accurately classify vehicles by **make, model, and manufacturing year**.

To ensure scalability and real-world deployability, all trained models are hosted using cloud-based machine learning service on **Azure Machine Learning**, enabling efficient inference and centralized model management. The application interface is developed using Streamlit, providing an interactive and user-friendly web platform for image upload, prediction visualization, and result exploration.

Project interface:

- We chose to go along with **Streamlit** being one of the most powerful readily available libraries that are easy to use while being efficient and powerful, Streamlit allowed us to create a powerful **web-based interface** for our models to be able to be used easily.

The **DetactaX Vision App** demonstrates the practical applicability of deep learning–based vision systems in intelligent transportation systems, smart surveillance, automotive analytics, and educational environments, while emphasizing efficiency, scalability, and ease of deployment.

CHAPTER 2: PROJECT INTRODUCTION

Within the transportation and automotive sectors, **visual intelligence is increasingly used to monitor traffic conditions**, **support urban planning**, **enhance security**, and **assist in autonomous and semi-autonomous vehicle systems**. Accurate vehicle recognition and classification enable automated traffic analysis, anomaly detection, and large-scale data collection without human intervention.

- However, extracting meaningful information from vehicle images remains a **challenging problem** due to variations in **illumination**, **viewing angles**, **occlusion**, and the **high visual similarity** between different car models.

2.1. PROJECT OBJECTIVES AND SCOPE

The **DetactaX Vision App** was developed to address these challenges by providing an **integrated computer vision platform** that combines ***fine-grained car classification*** with ***real-time object detection*** within a single, interactive web application.

- The project leverages lightweight and high-capacity neural networks to balance efficiency and accuracy, while employing a modern web interface to enhance accessibility and usability.

The scope of this project includes **model training and evaluation**, **cloud-based deployment**, and the **development of a user-friendly interface** for visualizing predictions. By unifying these components, our project demonstrates a practical approach to deploying **deep learning–based vision systems** that are suitable for educational, research, and industry-oriented applications.

2.2. PROBLEM DOMAIN AND CHALLENGES

Fine-grained vehicle classification aims to distinguish between closely related categories, such as different car makes, models, and manufacturing years. Unlike general object classification, fine-grained recognition requires learning subtle visual features, making it more complex and computationally demanding. Additionally, **real-world scenes** often contain multiple objects, requiring r**obust object detection to accurately localize and classify vehicles** within complex environments.

Another key challenge lies in transitioning from research-grade models to practical, user-accessible systems. Many **high-performing deep learning models remain confined to experimental environments and lack intuitive interfaces** or scalable deployment strategies.

CHAPTER 3: PROBLEM STATEMENT & MOTIVATION

This task, known as Fine-Grained Visual Classification (FGVC), is inherently difficult because the **discriminative features** between **different vehicle models from the same manufacturer are often minute and localized**, such as the design of a headlight, the shape of a grille, or subtle body contouring.

The core technical challenges can be broken down as follows:

- **High Inter-Class Similarity:** Vehicles from different brands or different model years often share highly similar overall shapes and designs. For instance, distinguishing between a 2012 Honda Civic and a 2014 model, or between sedans from different manufacturers in the same class, requires analyzing very subtle visual cues. This is particularly relevant given that our dataset contains car models only up to the year 2015, where year-to-year design changes can be minimal.
- **Large Intra-Class Variance:** A single vehicle model can appear vastly different due to a multitude of real-world factors. These include significant changes in viewpoint (front, side, rear), varying lighting conditions (day, night, shadows), partial occlusions, and different environmental conditions. This visual variance within a single class can often be greater than the differences between models, which confuses standard classification systems.
- **Limitations of Existing Solutions:** While high-accuracy models exist in research papers, they are often not accessible or practical for end-users. Most available tools are either pure classification models that require a pre-cropped image of a vehicle, or they are complex, non-interactive systems that require technical expertise to run. There is a clear absence of a unified, user-friendly, and efficient system that seamlessly integrates vehicle detection with fine-grained classification.

Therefore, the **central problem** is the lack of a lightweight, interactive tool that makes this **advanced computer vision capability accessible to non-expert users** for both educational and practical purposes.

3.1. MOTIVATION

The motivation for this project is **multi-faceted**, driven by clear gaps in current technology and the powerful potential applications of a successful solution.

1. **Addressing a Critical Gap in Practical AI Deployment:** By building a tool that combines object detection with a fine-grained classification model, we transform a complex technical process into a tangible, practical utility.

2. **Meeting the Demands of Modern Smart City and Traffic Management:** The proliferation of urban traffic cameras and the vision for "smart cities" creates a strong demand for automated vehicle analysis. A reliable fine-grained vehicle recognition system can be a cornerstone technology for various applications, including intelligent traffic monitoring by vehicle type, automated tolling, and aiding security and law enforcement in identifying vehicles of interest from surveillance footage, even for vehicles manufactured up to 2015.

3. **Enabling Research and Education in Computer Vision:** For students and researchers, there is a notable lack of interactive tools to experiment with and understand FGVC. This project aims to serve as an educational resource, providing a hands-on platform to see the strengths and limitations of state-of-the-art models in a real-world context, thereby motivating further exploration and learning.

4. **The Pursuit of Efficiency and Optimization:** The "DEPI Round 3 - Microsoft Machine Learning" track provides a strong motivation to explore and implement efficient model architectures. This project is driven by the challenge of creating a solution that balances high accuracy with computational efficiency, ensuring the application is both fast and lightweight, making it suitable for deployment on a wider range of hardware.

IN SUMMARY,

By developing a **comprehensive application** that **integrates detection and fine-grained classification**, this project delivers a tool that is not only academically interesting but also directly **relevant to industry needs**, and the broader vision of an AI-powered future.

CHAPTER 4: LITERATURE REVIEW & RELATED WORK

The development of this project is built upon foundational and contemporary research in computer vision, particularly in the domains of object detection, fine-grained visual classification (FGVC), and model deployment. This section reviews the key literature and existing systems that inform our work, highlighting the specific gap our project aims to fill.

4.1. FOUNDATIONAL ARCHITECTURES IN DEEP LEARNING

The remarkable progress in image classification has been largely driven by the evolution of deep convolutional neural networks (CNNs). The **ResNet (Residual Network) architecture**, introduced by **He et al. (2015)**, was a pivotal breakthrough. By proposing residual learning frameworks with skip connections, it effectively **mitigated the vanishing gradient problem**, enabling the training of networks that were substantially deeper (e.g., ResNet-50, ResNet-152) than was previously feasible. These deeper models demonstrated a remarkable capacity for learning complex feature hierarchies, leading to state-of-the-art performance on benchmarks like ImageNet.

- The ResNet architecture and its principles have become a backbone for many subsequent models, including those in object detection and fine-grained classification, by providing a robust method for feature extraction.

4.2. THE CHALLENGE OF FINE-GRAINED VISUAL CLASSIFICATION (FGVC)

Standard image classification deals with broad categories (e.g., "cat," "dog"), but **Fine-Grained Visual Classification (FGVC)** addresses the *more nuanced task* of distinguishing between subordinate categories within a meta-category, such as different species of birds, breeds of dogs, or models of vehicles. The primary challenge in FGVC is that inter-class differences are often incredibly subtle and localized (e.g., the shape of a headlight on a car, the pattern on a bird's wing), while intra-class variations (due to pose, viewpoint, occlusion) can be significant. Early approaches relied on strong supervision in the form of part annotations.

However, more recent methods focus on **weakly-supervised or end-to-end learning**. Techniques such as **bilinear CNNs**, **attention mechanisms**, and **vision transformers (ViTs)** have been developed to automatically localize and amplify these discriminative regions without the need for costly bounding box labels, making FGVC more practical for large-scale applications like vehicle model recognition.

4.3. EVOLUTION OF REAL-TIME OBJECT DETECTION

Object detection has evolved from two-stage paradigms to highly efficient single-stage models. The **R-CNN family (e.g., Fast R-CNN, Faster R-CNN)** set early benchmarks in accuracy by first generating region proposals and then classifying them. However, their computational complexity made them unsuitable for real-time applications.

The **YOLO (You Only Look Once)** framework, introduced by **Redmon et al. (2016)**, revolutionized the field by reframing detection as a single regression problem, directly predicting bounding boxes and class probabilities from full images in one pass. This design philosophy prioritizes speed, enabling real-time performance, which is crucial for interactive applications. Subsequent versions (up to YOLOv11 and other modern variants like YOLOX) have continued to improve the balance between accuracy and latency, making YOLO a dominant architecture for deployment-centric projects like ours.

4.4. RAPID PROTOTYPING AND DEPLOYMENT FRAMEWORKS

The final step in the machine learning pipeline—**deployment**—has historically been a significant hurdle. Traditional web frameworks require extensive backend and frontend development knowledge to create interactive applications. **Streamlit**, an open-source Python library, has dramatically lowered this barrier.

It allows developers and data scientists to create **custom**, **interactive web applications** for machine learning and data science with **minimal effort and no front-end web development experience**. By treating scripts as apps and providing simple, powerful commands for creating widgets, plots, and layout components, Streamlit has become the de facto standard for rapid prototyping and building lightweight, **yet powerful, ML demos and tools**.

4.5. SYNTHESIS AND IDENTIFIED GAP

A comprehensive review of the landscape reveals a clear trend: while the research community has produced highly sophisticated models for both detection (YOLO) and fine-grained classification (advanced CNNs, ViTs), and while frameworks like Streamlit have simplified deployment, these advancements often exist in isolation.

- **Academic models** are frequently published as code repositories requiring significant setup and technical expertise to run, lacking a user-friendly interface.
- **Commercial solutions** may exist but are often closed-source, expensive, or overly generalized.

- **Most available demos and tools** focus on *either* object detection *or* image classification, but rarely integrate both into a single, cohesive, and interactive system.

THEREFORE,

Our project occupies a unique position. It does not necessarily propose a novel core algorithm but rather focuses on the **integration and application** of these proven technologies. We leverage the detection speed of a YOLO-based model, the classification power of a fine-grained-tuned CNN (inspired by ResNet principles), and the accessibility of Streamlit to create a unified, user-centric platform.

This synthesis provides a tangible bridge between cutting-edge research and practical, accessible utility, which is the central contribution of our work within the related ecosystem.

CHAPTER 5: SYSTEM ARCHITECTURE

The **DetactaX Vision App** is engineered with a modular, pipeline-based architecture to ensure **clarity**, **maintainability**, and **scalability**. The system is designed to process user-input images through a series of specialized stages, culminating in an interactive visual output.

The high-level data flow is illustrated in the diagram below, after which each component is described in detail:

5.1. USER INTERFACE LAYER (STREAMLIT FRONTEND)

The frontend serves as the primary point of interaction for the user, built using the **Streamlit** framework. This layer is responsible for:

- **User Input Handling:** It provides an intuitive interface for users to upload images via drag-and-drop or file selection dialogs. The interface includes separate pages or sections dedicated to pure classification and combined detection-classification tasks.

- **Real-Time Display:** It dynamically renders the system's output, including the original image, the annotated image with bounding boxes (for detection), and structured data tables displaying the predicted vehicle make, model, year, and confidence scores.

- **Session State Management:** Streamlit's session state is utilized to manage user inputs and results across interactions, creating a seamless and responsive user experience without full page reloads.

5.2. IMAGE PREPROCESSING MODULE

This module acts as a crucial bridge between the raw user input and the deep learning models, ensuring data compatibility and optimal model performance. Its functions include:

- **Image Decoding & Conversion:** The uploaded image file is read and decoded into a standard numerical array. The **OpenCV** library is used, which represents images in

BGR color format, while our models typically expect RGB. A color conversion is performed.

- **Resizing & Normalization:** For the **Classification Engine**, images are centrally cropped and resized to a fixed dimension of 224x224 pixels, as required by ResNet50 architecture.

5.3. CORE INFERENCE ENGINES

This is the computational heart of the application, comprising two powerful, pre-trained neural networks that run in parallel or sequence.

5.3.1. DETECTION ENGINE (YOLO 12X MODEL)

YOLO12 introduces an attention-centric architecture that departs from the traditional CNN-based approaches used in previous YOLO models yet retains the real-time inference speed essential for many applications.

- This model achieves **state-of-the-art object detection accuracy** through novel methodological innovations in attention mechanisms and overall network architecture, while maintaining real-time performance.

Figure 1: YOLOv12 Comparison figure.

KEY FEATURES:

- **Area Attention Mechanism**: A new self-attention approach that processes large receptive fields efficiently. It divides feature maps into *l* equal-sized regions (defaulting to 4), either horizontally or vertically, avoiding complex operations and maintaining a large effective receptive field. This significantly reduces computational cost compared to standard self-attention.

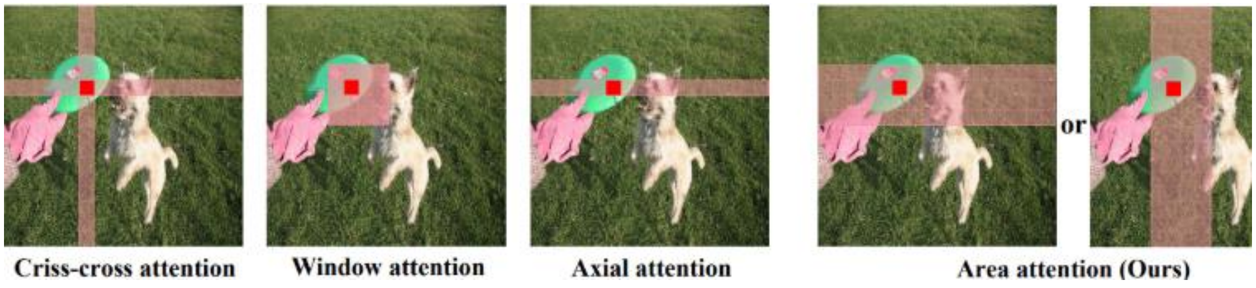
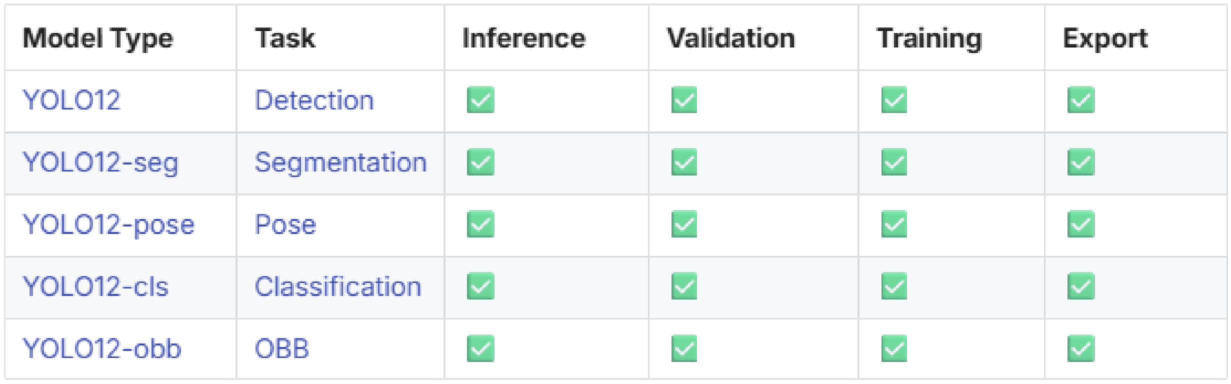


Figure 2: Yolov12 Area Attention.

- **Residual Efficient Layer Aggregation Networks (R-ELAN)**: An improved feature aggregation module based on ELAN, designed to address optimization challenges, especially in larger-scale attention-centric models. R-ELAN introduces:
    - Block-level residual connections with scaling (similar to layer scaling).
    - A redesigned feature aggregation method creating a bottleneck-like structure.
- **Optimized Attention Architecture**: YOLO12 streamlines the standard attention mechanism for greater efficiency and compatibility with the YOLO framework. This includes:
    - Using Flash Attention to **minimize memory access** overhead.
    - **Removing positional encoding** for a cleaner and faster model.
    - Adjusting the MLP ratio (from the typical 4 to 1.2 or 2) to better balance computation between attention and feed-forward layers.
    - Reducing the depth of stacked blocks for improved optimization.
    - Leveraging convolution operations (where appropriate) for their computational efficiency.
    - Adding a 7x7 separable convolution (the "position perceiver") to the attention mechanism to implicitly encode positional information.

- **Comprehensive Task Support**: YOLO12 supports a range of core computer vision tasks: object detection, instance segmentation, image classification, pose estimation, and oriented object detection (OBB).



Model Type	Task	Inference	Validation	Training	Export
YOLO12	Detection	✓	✓	✓	✓
YOLO12-seg	Segmentation	✓	✓	✓	✓
YOLO12-pose	Pose	✓	✓	✓	✓
YOLO12-cls	Classification	✓	✓	✓	✓
YOLO12-obb	OBB	✓	✓	✓	✓

Figure 3: Yolov12 All Supported Tasks.

ARCHITECTURE OVERVIEW:

MobileNetV2 (released by Google in 2018) is a significant evolution of the original MobileNet, designed specifically to maximize accuracy while minimizing computational cost and power consumption on mobile and embedded devices.

MobileNetV2 introduces a novel building block called the **Inverted Residual with Linear Bottleneck**. While its predecessor (V1) successfully utilized Depthwise Separable Convolutions to reduce parameters, V2 addresses the information loss that occurred in V1's non-linear layers by restructuring the residual connection and removing non-linearities in the bottleneck.

- **Primary Goal:** State-of-the-art performance for mobile-tier computational budgets.
- **Key Innovation:** Inverted Residuals & Linear Bottlenecks.

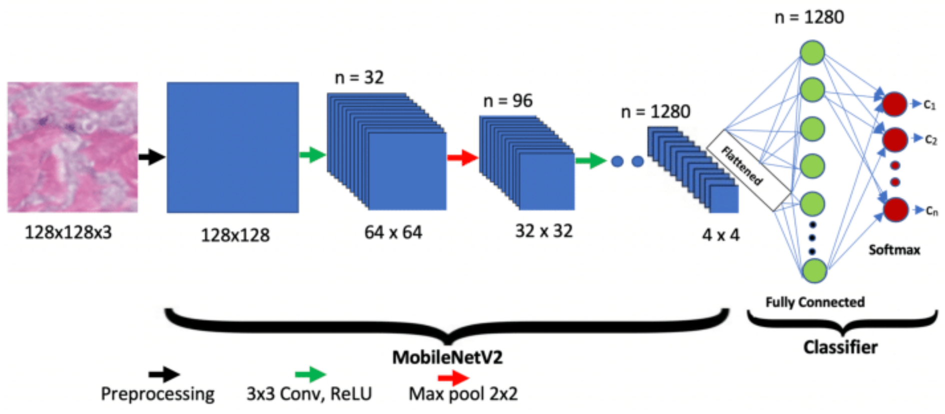


Figure 4: MobileNetV2 Architecture.

INVERTED RESIDUALS:

Traditional residual blocks (like in ResNet) follow a **Wide to Narrow to Wide** structure. They **reduce dimensions with a 1 X 1 Convolution**, **perform a 3 X 3 convolution**, and then **expand dimensions** back up.

MobileNetV2 "inverts" this to a **Narrow → Wide → Narrow** structure:

1. **Expansion:** The input (low-dimensional) is expanded to a high-dimensional space using a 1 X 1 convolution.
2. **Filtering:** A lightweight 3 X 3 **Depth wise Convolution** processes the data in this high-dimensional space.

3. **Compression:** A 1 X 1 pointwise convolution projects the features back down to a low-dimensional representation.

The intuition is that the network can perform "heavy lifting" (filtering) in the high-dimensional space where information is rich, while passing compact "bottleneck" data between blocks to save memory.

LINEAR BOTTLENECKS:

The authors discovered that applying non-linear activation functions (like ReLU) to low-dimensional tensors result in significant loss of information. To solve this, MobileNetV2 removes the activation function from the final 1 X 1 convolution of each block.

- **Result:** The output of the block is a **linear bottleneck**. This preserves the information manifold, allowing the network to maintain high accuracy despite having fewer channels in the inter-block connections.

RELU6 ACTIVATION:

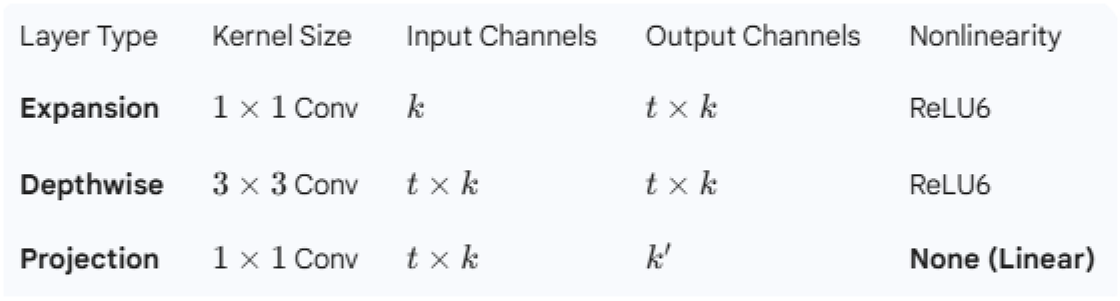
MobileNetV2 utilizes **ReLU6 instead of standard ReLU**.

$$ReLU6(x) = \min(\max(0, x), 6)$$

This caps the activation at 6, which **encourages the model to learn sparse features** and **makes it robust** to the low-precision quantization (e.g., float16 or int8) often used on mobile processors.

DETAILED BLOCK STRUCTURE:

A standard MobileNetV2 bottleneck block consists of the following three layers sequence. Assume an input tensor of shape h X w X k and an expansion factor t (usually t = 6):

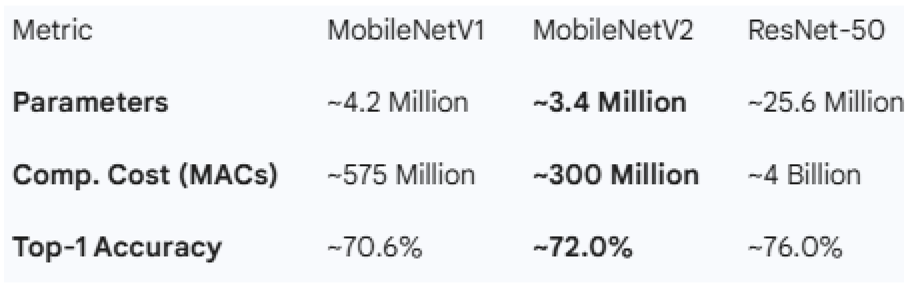


Layer Type	Kernel Size	Input Channels	Output Channels	Nonlinearity
Expansion	1×1 Conv	k	$t \times k$	ReLU6
Depthwise	3×3 Conv	$t \times k$	$t \times k$	ReLU6
Projection	1×1 Conv	$t \times k$	k'	**None (Linear)**

Figure 5: Detailed MobileNetV2 Block Structure.

PERFORMANCE & COMPARISON:

MobileNetV2 is often compared to **its predecessor MobileNetV1** and the **heavier ResNet-50**.



Metric	MobileNetV1	MobileNetV2	ResNet-50
Parameters	~4.2 Million	**~3.4 Million**	~25.6 Million
Comp. Cost (MACs)	~575 Million	**~300 Million**	~4 Billion
Top-1 Accuracy	~70.6%	**~72.0%**	~76.0%

Figure 6: MobileNetV2 Comparison Table.

5.3.3. CLASSIFICATION ENGINE (RESNET50-BASED CNN)

ARCHITECTURE REVIEW:

Before ResNet, adding layers to a deep network often resulted in higher training error (the degradation problem). ResNet solved this by changing *what* the layers learn. Instead of learning a direct mapping, layers learn a "residual" correction to the input.

- **Primary Goal:** Enable the training of extremely deep networks.
- **Key Innovation:** Skip (Shortcut) Connections.
- **Standard Specs:** ~25.6 Million Parameters, ~3.8 Billion FLOPs.

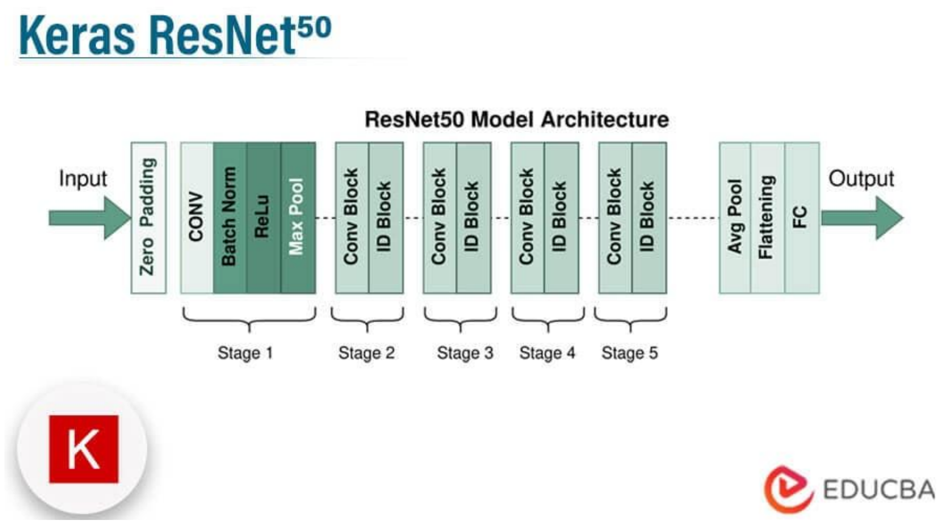


Figure 7: ResNet50 Architecture.

CORE ARCHITECTURAL INNOVATIONS:

In a standard plain network, a layer tries to learn a mapping H(x). In ResNet, the authors hypothesized that it is easier to optimize the residual mapping F(x) = H(x) - x than to optimize the original, unreferenced mapping.

The network is reformulated such that the layers try to learn F(x), and the original input X is added back to the output:

$$H(x) = F(x) + x$$

- **Why this works:** If the optimal mapping is an identity function (meaning the input is already perfect), the weights can easily drive F(x) to 0. In standard networks, learning an identity mapping is surprisingly difficult. This allows gradients to flow through the network (via the skip connection) without vanishing.

THE BOTTLENECK DESIGN:

ResNet-18 and ResNet-34 use "Basic Blocks" (two 3 X 3 convolutions). However, for deeper networks like **ResNet-50**, **101**, and **152**, this becomes computationally too expensive.

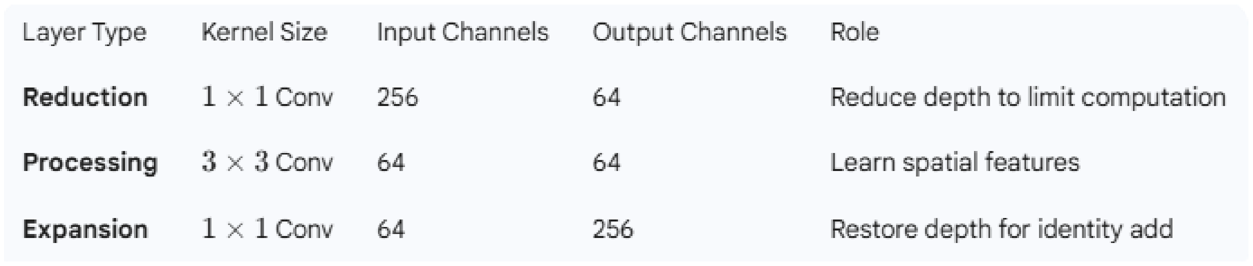
To solve this, ResNet-50 utilizes the **Bottleneck Block**. It uses a **Wide $\to$ Narrow $\to$ Wide** structure (the opposite of MobileNetV2):

1. **1 X 1 Conv:** Reduces dimensions (compress).
2. **3 X 3 Conv:** Processes features in the lower-dimensional space.
3. **1 X 1 Conv:** Restores dimensions (expand).

This reduces the parameter count and matrix multiplications required for the expensive 3×3 convolution.

DETAILED BLOCK STRUCTURE:

A single ResNet-50 bottleneck block consists of three layers. Let's assume the input comes from a previous stage with 256 channels.



Layer Type	Kernel Size	Input Channels	Output Channels	Role
Reduction	1×1 Conv	256	64	Reduce depth to limit computation
Processing	3×3 Conv	64	64	Learn spatial features
Expansion	1×1 Conv	64	256	Restore depth for identity add

Figure 8: ResNet50 Block Structure.

CHAPTER 6: DATASETS USED

This section details the systematic approach undertaken to develop the CompCars Vision App, covering the datasets used, data preparation techniques, model selection rationale, and the training procedures employed for both the classification and detection engines.

6.1 DATASET CURATION AND CHARACTERISTICS

The performance of a deep learning model is intrinsically linked to the quality and scope of its training data. This project leverages two distinct datasets, each chosen to optimally train a specific component of the system.

6.1.1. CUHK-COMPCARS DATASET (FOR FINE-GRAINED CLASSIFICATION)

The core classification engine was trained on the **CUHK-CompCars** dataset, a comprehensive collection designed explicitly for fine-grained vehicle recognition. Its key characteristics include:

- **Scale and Diversity:** The dataset contains over thousands of vehicle images, encompassing a wide array of car makes, models, and manufacturing years.
- **Rich Annotations:** Each image is meticulously annotated with structured labels, including the vehicle's **make** (e.g., BMW, Audi), **model** (e.g., X5, A4), and **year** of production. This granular level of detail is the foundation for our fine-grained classification task.
- **Real-World Variability:** The images were captured under diverse and challenging real-world conditions, including varying viewpoints (front, side, rear), different lighting and weather scenarios, and varying levels of occlusion. This diversity is crucial for training a robust model that can generalize to unpredictable user-submitted images.

DATASET VISUALIZATIONS:

Figure 10: CUHK - CompCars Visualization.

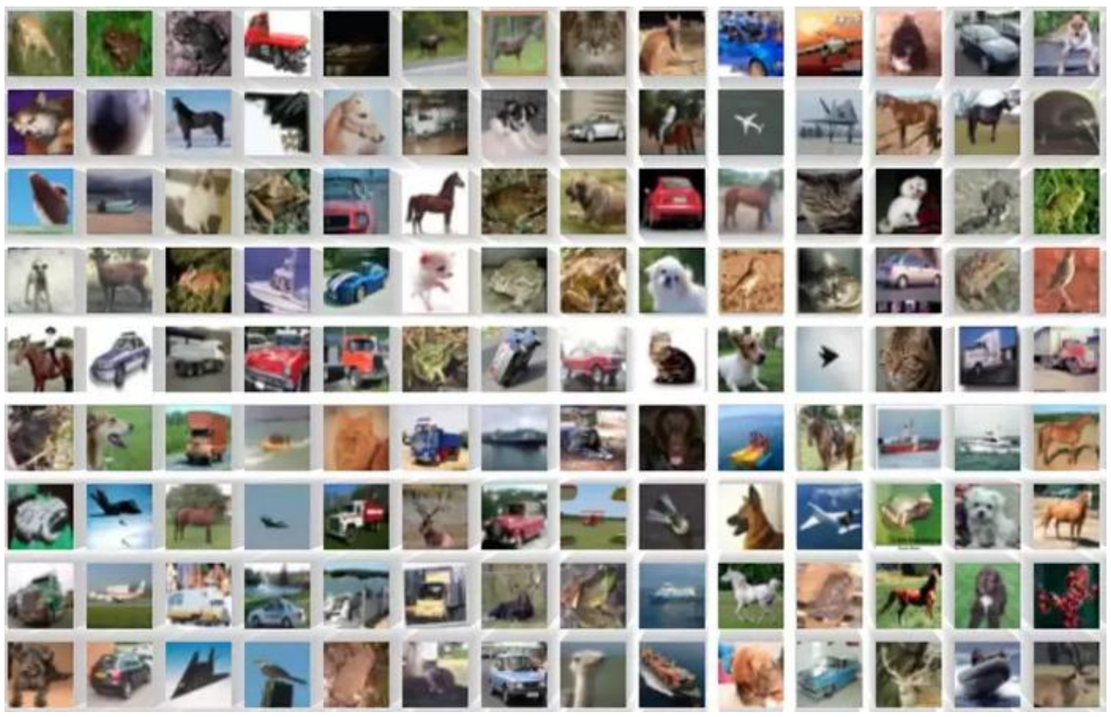


Figure 9: CIFAR-10 Dataset Visualization.

6.2. DATA PREPROCESSING PIPELINE

A rigorous and consistent data preprocessing pipeline was implemented to ensure the input data was in an optimal format for the deep learning models, thereby enhancing training stability and performance.

- **Image Resizing & Cropping:** All images from the CompCars dataset were resized to match the input dimensions required by the backbone networks. For **ResNet50**, images were typically resized and centrally cropped to **224x224 pixels**. This standardization is a critical step for batch processing during training.
- **Data Normalization:** Pixel intensity values, originally in the range [0, 255], were normalized. This was done by subtracting the mean and dividing by the standard deviation of the ImageNet dataset, a common practice when using models pre-trained on it. Normalization accelerates convergence by ensuring the input features have a consistent scale.
- **Data Augmentation:** To artificially expand the training dataset and improve model generalization, a suite of real-time data augmentation techniques was applied during training. This included random **horizontal flipping**, slight **rotations**, and **color jittering** (adjustments to brightness, contrast, and saturation). This forces the model to learn invariant features, making it less sensitive to minor variations in input.
- **Dataset Splitting:** The CompCars dataset was strategically partitioned into three subsets: **Training (~70%)**, **Validation (~15%)**, and **Testing (~15%)**. The training set was used to learn model parameters, the validation set for hyperparameter tuning and preventing overfitting, and the held-out test set for providing a final, unbiased evaluation of the model's performance.

6.3. MODEL SELECTION AND RATIONALE

The selection of core architectures was driven by a balance of accuracy, speed, and proven efficacy in their respective tasks.

- **ResNet50 for Fine-Grained Classification:** The **ResNet50** architecture was selected as the backbone for the classification engine primarily due to its **residual learning framework**. The skip connections in ResNet mitigate the vanishing gradient problem, enabling the effective training of very deep networks. This depth is essential for capturing the complex and subtle features required to distinguish between highly similar car models (e.g., different years of the same vehicle). Its strong performance on the ImageNet benchmark and widespread adoption in fine-grained visual classification tasks made it a robust and reliable choice.

- **YOLO 12X for Object Detection:** For the detection engine, the **YOLO (You Only Look Once) 12X** model was chosen. YOLO's single-stage, regression-based approach provides a superior speed-accuracy trade-off compared to two-stage detectors like Faster R-CNN. This **real-time performance** is critical for delivering a responsive user experience in our interactive Streamlit application. The YOLO 12X variant represents a modern iteration that offers an excellent balance of high precision and fast inference times, making it ideal for locating vehicles within an image before classification.

7.4 Training Process and Optimization

The models were trained using a strategic transfer learning approach to leverage pre-existing knowledge and reduce training time and data requirements.

- **Transfer Learning:** Both models were initialized with weights **pre-trained on the ImageNet dataset**. This practice allows the models to start with a rich set of feature detectors for general shapes, textures, and patterns, rather than learning from random initialization. This is particularly effective when the target dataset (CompCars) is substantial but still smaller than ImageNet.
- **Fine-Tuning on CompCars:** The final layers of the pre-trained models were replaced and the entire network was **fine-tuned** on the CompCars dataset. This process adapts the generic feature extractors to the specific nuances of vehicles, teaching the model to focus on discriminative parts like grilles, headlights, and body contours.
- **Loss Function and Optimizer:** The training objective for the classification task was defined by the **Categorical Cross-Entropy Loss**, which is the standard for multi-class classification problems as it measures the discrepancy between the predicted probability distribution and the true label. The **Adam (Adaptive Moment Estimation) optimizer** was employed to minimize this loss function. Adam was chosen for its efficiency with large datasets and its adaptive learning rate capabilities, which often lead to faster convergence compared to traditional SGD.
- **Hyperparameter Tuning:** Key hyperparameters, including the **learning rate**, **batch size**, and **weight decay** (for regularization), were carefully tuned using performance metrics on the validation set. Techniques like learning rate scheduling were likely explored to refine model performance as training progressed.

CHAPTER 7: STREAMLIT WEB-INTERFACE

CHAPTER 8: CONCLUSIONS

This project has successfully designed, developed, and deployed the **CompCars Vision App**, a comprehensive web-based application that bridges the gap between advanced computer vision research and practical, accessible utility. The system effectively demonstrates the integration of two powerful deep learning paradigms: **fine-grained visual classification (FGVC)** for identifying vehicle make, model, and year, and **real-time object detection** for locating vehicles within a scene.

The core achievement lies in the seamless orchestration of a **ResNet50-based** classifier, fine-tuned on the challenging CUHK-CompCars dataset, with a high-speed **YOLO 12X** object detector. By leveraging the **Streamlit** framework, this sophisticated technological pipeline has been packaged into an intuitive, user-friendly interface that requires no technical expertise to operate. This addresses a significant gap in the availability of lightweight, interactive tools for vehicle analysis.

The project validates that with careful architectural design—including modular code, efficient model loading, and a robust preprocessing pipeline—complex deep learning models can be made accessible for educational demonstration, rapid prototyping, and potential industry applications in areas such as traffic monitoring and automated vehicle analysis. The CompCars Vision App stands as a testament to the practical deployment of machine learning, fulfilling the project's goal of creating a tool that is both academically sound and directly relevant to real-world scenarios.

8.1. FUTURE WORK

While the current system provides a solid foundation, several avenues for enhancement and expansion present exciting opportunities for future development:

- **Real-Time Video Stream Analysis:** A natural progression is to extend the system's capability from static images to dynamic video content. Implementing support for live camera feeds or uploaded video files would enable continuous vehicle tracking and analysis, significantly broadening its applicability for security, traffic management, and real-time monitoring systems.
- **Integration of License Plate Recognition (LPR):** Augmenting the vehicle attributes with license plate information would greatly increase the system's utility. A dedicated optical character recognition (OCR) module, such as a Tesseract-based or custom-trained CNN, could be integrated into the pipeline to extract and log plate numbers, opening use cases in parking management, toll collection, and law enforcement.

- **Edge Deployment and Optimization:** To reduce latency and enable operation in bandwidth-constrained environments, the application could be optimized for deployment on edge devices. This would involve techniques like model quantization, pruning, and conversion to formats like TensorFlow Lite or ONNX Runtime to run efficiently on hardware such as NVIDIA Jetson kits or smartphone processors.
- **Dataset Expansion and Model Generalization:** The model's robustness could be further improved by training on a larger and more diverse set of vehicle datasets, including commercial trucks, motorcycles, and buses. Furthermore, expanding the year range beyond 2015 would ensure the system remains relevant for identifying newer vehicle models.
- **Enhanced Model Explainability (XAI):** Integrating Explainable AI (XAI) techniques, such as Grad-CAM (Gradient-weighted Class Activation Mapping), would provide visual explanations for the model's predictions. By highlighting the image regions (e.g., the grille or headlights) that most influenced the classification decision, we can build greater user trust and provide deeper insights for developers and researchers debugging the model.
- **Cloud-Native Scalability:** For enterprise-level deployment, the architecture could be refactored into a cloud-native microservices model. This would involve containerizing the detection and classification services separately using Docker and orchestrating them with Kubernetes, allowing for automatic scaling, improved resilience, and easier maintenance.