Car Type Classification: A Deep Learning Approach

Course: AI335 Neural Networks & Deep Learning
Module: Spring Semester 1 “2025-2026”
Institution: Helwan University - Faculty of Computers & Artificial Intelligence
Date: December 9, 2025

Executive Summary

This document presents a comprehensive analysis of a deep learning project focused on fine-grained car type classification using the Stanford Cars dataset. The project implements and evaluates four distinct convolutional neural network (CNN) architectures: **ResNet-50**, **VGG-19**, **Inception V1 (GoogLeNet)**, and **MobileNetV2**. Through systematic experimentation across varying task complexities (10, 20, and 196 classes), this study demonstrates the effectiveness of transfer learning and architectural innovations in addressing the challenges of fine-grained image classification.

The experimental results reveal that **ResNet-50** achieves the highest performance on the full 196-class dataset with 84.06% accuracy, significantly outperforming other models. This success is attributed to its residual learning framework, which enables deep feature extraction while maintaining stable gradient flow. The comparative analysis provides insights into the strengths and limitations of each architecture, offering practical guidance for model selection in similar computer vision tasks.

---

1. Introduction

Fine-grained image classification represents one of the most challenging problems in computer vision, requiring models to distinguish between visually similar categories that differ only in subtle details. The task of classifying car makes and models

exemplifies this challenge, as vehicles within the same category often share similar shapes, colors, and structural features, with differences limited to minor design elements such as grilles, headlights, or body contours.

This project addresses the car type classification problem using the **Stanford Cars dataset**, which contains 16,185 images across 196 classes of cars. The dataset encompasses a diverse range of vehicles from different manufacturers, years, and models, providing a robust benchmark for evaluating fine-grained classification algorithms. Each class represents a specific make, model, and year combination (e.g., "2012 Tesla Model S" or "2012 BMW M3 Coupe"), making the classification task particularly demanding.

To tackle this challenge, four state-of-the-art CNN architectures were selected, each representing different design philosophies and computational trade-offs:

1. **ResNet-50**: A deep residual network that uses skip connections to enable training of very deep models
2. **VGG-19**: A simple yet deep architecture trained from scratch to evaluate performance without pre-trained weights
3. **Inception V1 (GoogLeNet)**: A multi-scale feature extraction network with parallel convolutional branches
4. **MobileNetV2**: A lightweight architecture designed for mobile and embedded devices using depthwise separable convolutions

The project employs transfer learning for ResNet-50, Inception V1, and MobileNetV2, leveraging pre-trained weights from ImageNet to accelerate training and improve generalization. In contrast, VGG-19 was trained from scratch to provide a baseline for comparison. Experiments were conducted on three different subsets of the data (10, 20, and 196 classes) to analyze how model performance scales with task complexity.

This documentation provides detailed explanations of each architecture, presents comprehensive experimental results with visualizations, and offers a comparative analysis highlighting the strengths and weaknesses of each approach. The goal is to deliver clear, professional, and actionable insights for researchers and practitioners working on similar fine-grained classification problems.

---

2. Dataset and Preprocessing

2.1. Stanford Cars Dataset

The Stanford Cars dataset is a widely-used benchmark for fine-grained visual categorization. It contains 16,185 images of 196 classes of cars, split into 8,144 training images and 8,041 testing images. The classes are organized by make, model, and year, spanning vehicles from 1991 to 2012. The dataset presents several challenges:

- **High intra-class variability**: Images of the same car model may vary significantly in viewpoint, lighting, background, and occlusion
- **Low inter-class variability**: Different car models may appear visually similar, differing only in subtle design details
- **Imbalanced distribution**: Some classes have more training examples than others
- **Real-world conditions**: Images are captured in diverse environments with varying quality and resolution

2.2. Data Preprocessing

Proper preprocessing is essential for achieving optimal model performance. The following preprocessing steps were applied consistently across all experiments:

1. **Image Resizing**: All images were resized to match the input requirements of each architecture (224×224 for ResNet-50, VGG-19, and MobileNetV2; 299×299 for Inception V1)

2. **Normalization**: Pixel values were normalized using ImageNet statistics (mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225]) to align with the pre-trained model expectations

3. **Data Augmentation**: To improve model generalization and address class imbalance, the following augmentation techniques were applied during training:

    - Random horizontal flipping
    - Random rotation (±15 degrees)
    - Random brightness and contrast adjustment

- Random cropping and resizing

4. **Train-Test Split**: The official dataset split was maintained, with approximately 50% of images used for training and 50% for testing

2.3. Experimental Configurations

To analyze model performance under varying levels of complexity, three experimental configurations were designed:

Configuration	Number of Classes	Purpose
10-Class	10	Simplified task to evaluate baseline performance
20-Class	20	Medium complexity to assess scalability
196-Class	196	Full dataset for comprehensive evaluation

For the 10-class and 20-class experiments, a balanced subset of the most common car models was selected to ensure fair comparison across models.

---

3. Documentation of Architectures

This section provides detailed explanations of the four deep learning architectures implemented in this project. Each model represents a distinct approach to feature extraction and classification, with unique strengths and trade-offs.

3.1. ResNet-50

The Residual Network (ResNet) architecture, introduced by He et al. in 2016 [1], revolutionized deep learning by solving the degradation problem in very deep neural networks. The key innovation is the **residual learning framework**, which uses skip connections to enable identity mapping.

3.1.1. Core Concept: Residual Learning

Instead of learning a direct mapping H(x) from input x to output, ResNet learns a residual function F(x) = H(x) - x. The original mapping is then reconstructed as H(x) =

F(x) + x. This reformulation makes it easier for the network to learn identity mappings, which is crucial for training very deep networks.

> *"We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth." [1]*

3.1.2. Architecture Details

ResNet-50 consists of 50 layers organized into the following structure:

1. **Initial Convolution Block**:
    - 7×7 convolution with 64 filters, stride 2
    - Batch normalization and ReLU activation
    - 3×3 max pooling with stride 2
    - Reduces spatial dimensions from 224×224 to 56×56
2. **Residual Stages** (4 stages with bottleneck blocks):
    - **Stage 1 (conv2_x)**: 3 residual blocks, 64 filters, output size 56×56
    - **Stage 2 (conv3_x)**: 4 residual blocks, 128 filters, output size 28×28
    - **Stage 3 (conv4_x)**: 6 residual blocks, 256 filters, output size 14×14
    - **Stage 4 (conv5_x)**: 3 residual blocks, 512 filters, output size 7×7
3. **Classification Head**:
    - Global average pooling (reduces 7×7×2048 to 2048)
    - Fully connected layer with 196 outputs (one per class)
    - Softmax activation for probability distribution

3.1.3. Bottleneck Design

ResNet-50 uses a bottleneck design for efficiency, where each residual block contains three convolutional layers:

- **1×1 convolution**: Reduces channel dimensions (e.g., 256 → 64)
- **3×3 convolution**: Performs spatial feature extraction

- **1×1 convolution**: Restores channel dimensions (e.g., 64 → 256)

This design significantly reduces the number of parameters compared to using only 3×3 convolutions.

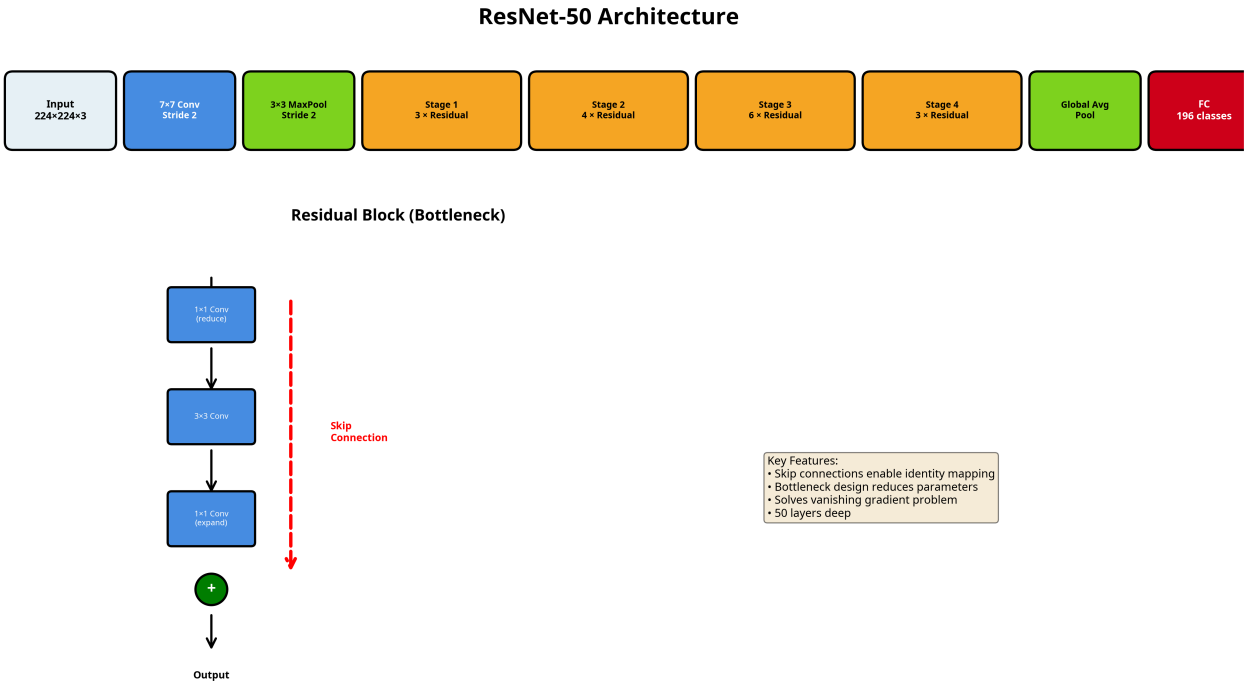

Figure 1: ResNet-50 architecture showing the residual learning framework with skip connections and bottleneck design.

3.1.4. Key Advantages

- **Solves degradation problem**: Skip connections enable training of very deep networks (50+ layers)
- **Stable gradient flow**: Direct paths for gradients prevent vanishing gradient issues
- **Effective transfer learning**: Pre-trained on ImageNet provides rich feature representations
- **Hierarchical features**: Deep structure learns features from simple edges to complex object parts

---

3.2. VGG-19

The VGG architecture, developed by Simonyan and Zisserman in 2014 [2], is characterized by its simplicity and uniformity. VGG-19 uses only 3×3 convolutional filters throughout the network, demonstrating that depth is more important than filter size for achieving high performance.

3.2.1. Design Philosophy

VGG-19 follows a straightforward design principle: stack small convolutional filters (3×3) to build a deep network. This approach has several advantages:

- Two 3×3 convolutions have the same receptive field as one 5×5 convolution but with fewer parameters
- Multiple non-linear activation functions increase model capacity
- Uniform architecture simplifies implementation and analysis

3.2.2. Architecture Details

VGG-19 consists of 19 layers with learnable weights (16 convolutional layers and 3 fully connected layers):

1. **Block 1**: 2 × (3×3 conv, 64 filters) + MaxPool
2. **Block 2**: 2 × (3×3 conv, 128 filters) + MaxPool
3. **Block 3**: 4 × (3×3 conv, 256 filters) + MaxPool
4. **Block 4**: 4 × (3×3 conv, 512 filters) + MaxPool
5. **Block 5**: 4 × (3×3 conv, 512 filters) + MaxPool
6. **Fully Connected Layers**: FC-4096, FC-4096, FC-196
7. **Output**: Softmax activation

Each convolutional layer is followed by ReLU activation, and each max pooling layer uses a 2×2 filter with stride 2, halving the spatial dimensions.

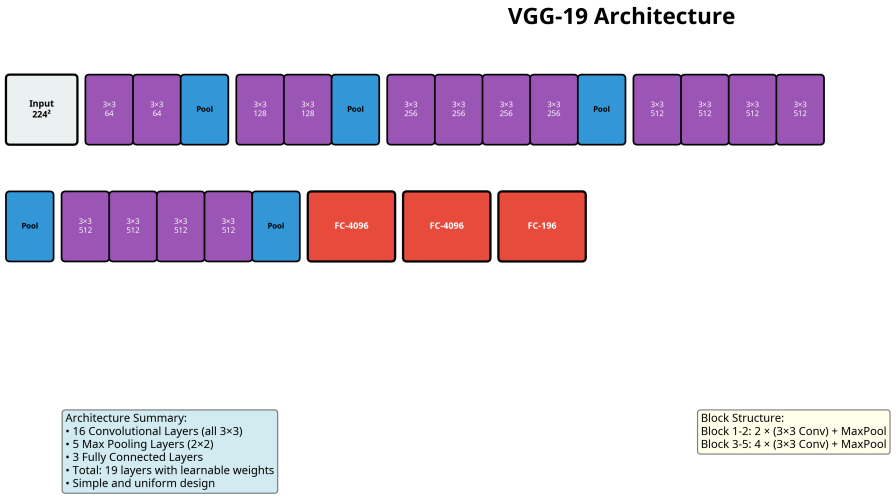

Figure 2: VGG-19 architecture showing the uniform structure with 3×3 convolutional layers organized in five blocks.

3.2.3. Training from Scratch

In this project, VGG-19 was trained from scratch (without pre-trained weights) to evaluate its performance in a data-limited scenario. This approach provides insights into:

- The importance of transfer learning for deep networks
- The challenges of training very deep networks without initialization
- The impact of architectural design on trainability

3.2.4. Key Characteristics

- **Simple and uniform**: Easy to understand and implement
- **Deep architecture**: 19 layers enable learning of complex features
- **Large parameter count**: Approximately 144 million parameters (prone to overfitting)
- **Computationally expensive**: Requires significant memory and training time

---

3.3. Inception V1 (GoogLeNet)

The Inception architecture, introduced by Szegedy et al. in 2015 [3], takes a fundamentally different approach by performing multi-scale feature extraction through parallel convolutional branches. This design enables the network to capture features at different scales simultaneously while maintaining computational efficiency.

3.3.1. Core Concept: Inception Module

The Inception module is the building block of the architecture. Instead of choosing a single filter size (e.g., 3×3 or 5×5), the module applies multiple filter sizes in parallel and concatenates their outputs:

- **Branch 1**: 1×1 convolution (captures point-wise features)
- **Branch 2**: 1×1 convolution → 3×3 convolution (medium-scale features)
- **Branch 3**: 1×1 convolution → 5×5 convolution (large-scale features)
- **Branch 4**: 3×3 max pooling → 1×1 convolution (pooling features)

The 1×1 convolutions before the 3×3 and 5×5 convolutions serve as "bottleneck" layers, reducing the number of input channels and thereby decreasing computational cost.

3.3.2. Architecture Details

Inception V1 (GoogLeNet) consists of 22 layers organized as follows:

1. **Initial Layers**:
    - 7×7 convolution, stride 2
    - 3×3 max pooling, stride 2
    - 3×3 convolution
2. **Inception Modules** (9 total, organized in 3 stages):
    - **Stage 1**: 2 Inception modules (large spatial dimensions, global features)
    - **Stage 2**: 5 Inception modules (medium dimensions, mid-level features)
    - **Stage 3**: 2 Inception modules (small dimensions, high-level features)

3. **Auxiliary Classifiers**: Two auxiliary classifiers are attached to intermediate layers during training to combat vanishing gradients and provide additional regularization

4. **Classification Head**:
    - Global average pooling
    - Dropout (40%)
    - Fully connected layer with 196 outputs
    - Softmax activation

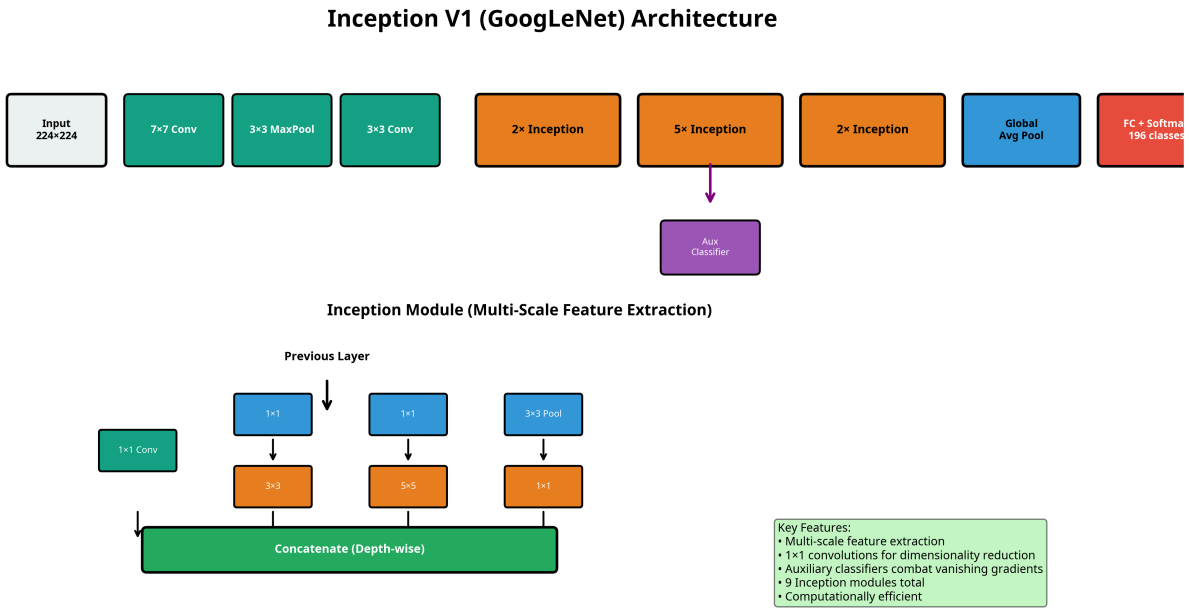

Figure 3: Inception V1 (GoogLeNet) architecture showing multi-scale feature extraction through parallel convolutional branches.

3.3.3. Computational Efficiency

Despite its complexity, Inception V1 is computationally efficient due to:

- **Dimensionality reduction**: 1×1 convolutions reduce channel dimensions before expensive operations

- **Global average pooling**: Replaces fully connected layers, reducing parameters
- **Factorized convolutions**: Smaller filters (1×1, 3×3, 5×5) instead of large filters

3.3.4. Key Advantages

- **Multi-scale feature extraction**: Captures features at different granularities
- **Computational efficiency**: Fewer parameters than VGG despite similar depth
- **Auxiliary classifiers**: Help with gradient flow during training
- **Flexibility**: Network can choose appropriate filter sizes for different features

---

3.4. MobileNetV2

MobileNetV2, introduced by Sandler et al. in 2018 [4], is designed specifically for mobile and embedded vision applications where computational resources are limited. The architecture introduces two key innovations: **inverted residual blocks** and **linear bottlenecks**.

3.4.1. Core Concepts

Depthwise Separable Convolutions: MobileNetV2 factorizes standard convolutions into two operations:

1. **Depthwise convolution**: Applies a single filter to each input channel separately (3×3 spatial filtering)
2. **Pointwise convolution**: Uses 1×1 convolutions to combine the outputs (channel-wise combination)

This factorization reduces computational cost by 8-9× compared to standard convolutions.

Inverted Residual Structure: Unlike ResNet, which reduces channels in the middle of the block (wide → narrow → wide), MobileNetV2 expands channels in the middle (narrow → wide → narrow). This design is more efficient for low-dimensional representations.

Linear Bottlenecks: The final 1×1 convolution in each block uses a linear activation (no ReLU) to preserve information in low-dimensional spaces, as ReLU can destroy information when operating on low-dimensional data.

3.4.2. Architecture Details

MobileNetV2 consists of:

1. **Initial Convolution**: 3×3 convolution with stride 2
2. **Inverted Residual Blocks** (19 blocks with varying configurations):
    - Expansion factor: 6× (channels are expanded by a factor of 6)
    - Stride: 1 or 2 (for downsampling)
    - Activation: ReLU6 (output = min(max(0, x), 6))
3. **Final Convolution**: 1×1 convolution to increase channels
4. **Classification Head**:
    - Global average pooling
    - Fully connected layer with 196 outputs
    - Softmax activation

Each inverted residual block follows this structure:

1. **1×1 Expansion**: Expands channels by 6× with ReLU6 activation
2. **3×3 Depthwise**: Spatial feature extraction with ReLU6 activation
3. **1×1 Projection**: Projects back to original channel count with linear activation
4. **Skip Connection**: Added when stride = 1 and input/output dimensions match

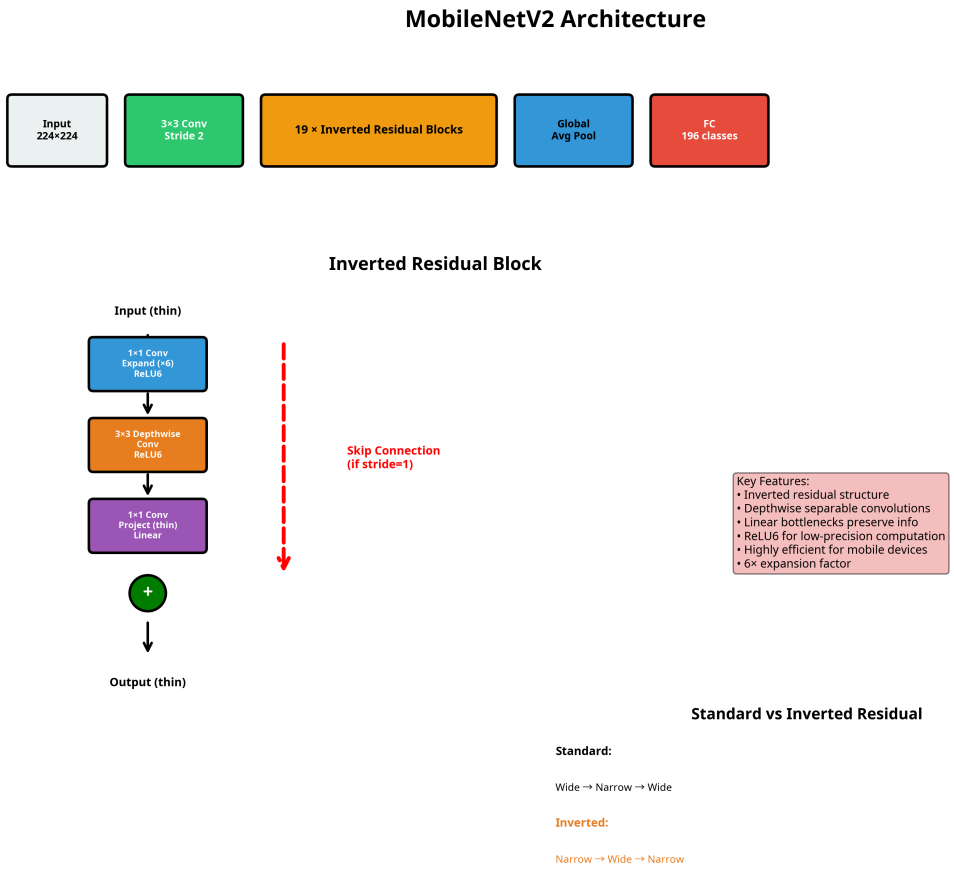

Figure 4: MobileNetV2 architecture showing inverted residual blocks with depthwise separable convolutions.

3.4.3. ReLU6 Activation

ReLU6 is defined as: $f(x) = \min(\max(0, x), 6)$

This activation function is used to prevent numerical instability in low-precision computations (e.g., on mobile devices) by capping the output at 6.

3.4.4. Key Advantages

- **Highly efficient**: 6-7× fewer parameters than ResNet-50
- **Mobile-friendly**: Designed for resource-constrained devices
- **Inverted residuals**: Better for low-dimensional feature spaces
- **Linear bottlenecks**: Preserve information in compressed representations

---

4. Comparative Analysis of Models

This section presents a comprehensive comparison of the four architectures based on their experimental performance across different task complexities. The analysis includes quantitative metrics, visualizations, and qualitative insights into the strengths and weaknesses of each model.

4.1. Performance on 196 Classes (Full Dataset)

The following table summarizes the performance of each model on the complete Stanford Cars dataset with all 196 classes:

Model	Accuracy	Precision (Macro)	Recall (Macro)	F1-Score (Macro)	AUC (Macro)
ResNet-50	**84.06%**	**84.78%**	**83.99%**	**83.98%**	**99.84%**
MobileNetV2	63.82%	66.85%	63.88%	63.82%	99.22%
VGG-19 (Scratch)	46.81%	49.91%	46.78%	46.68%	98.03%
Inception V1	38.49%	48.29%	38.29%	38.62%	96.49%

Table 1: Performance metrics for all models on the 196-class Stanford Cars dataset.

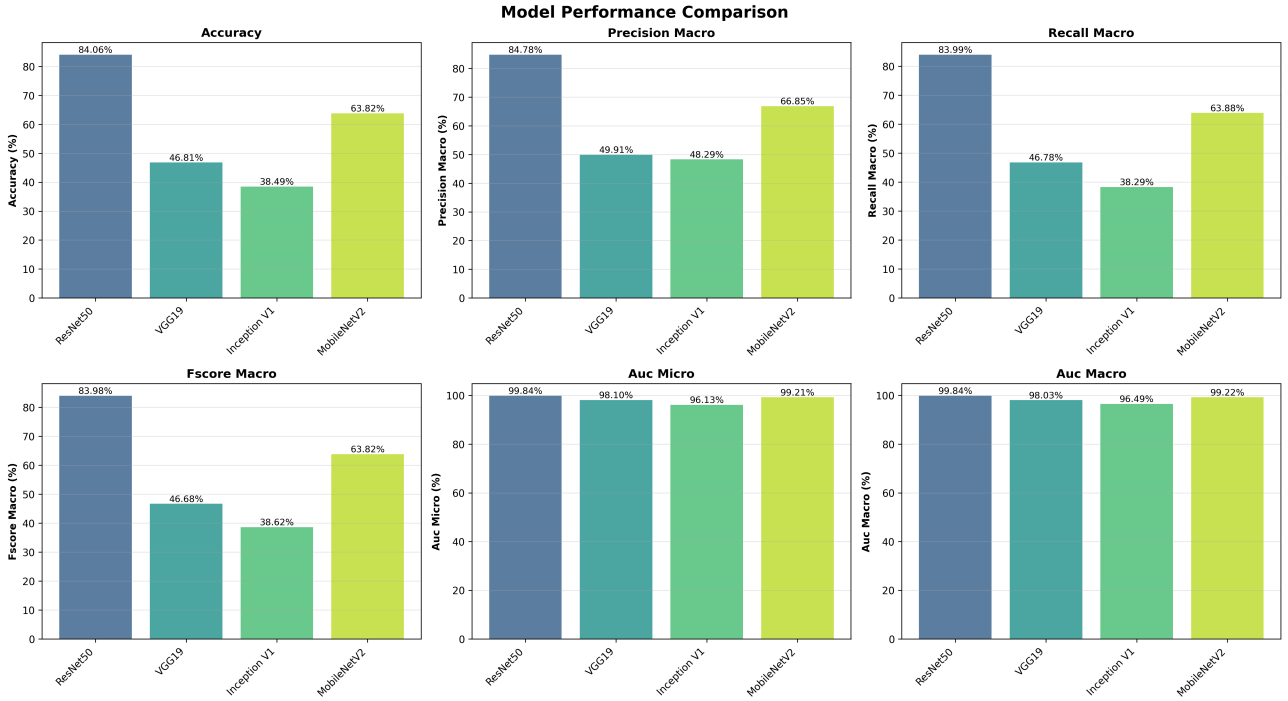

Figure 5: Comparative performance of all models across key metrics on the 196-class dataset. ResNet-50 significantly outperforms other architectures.

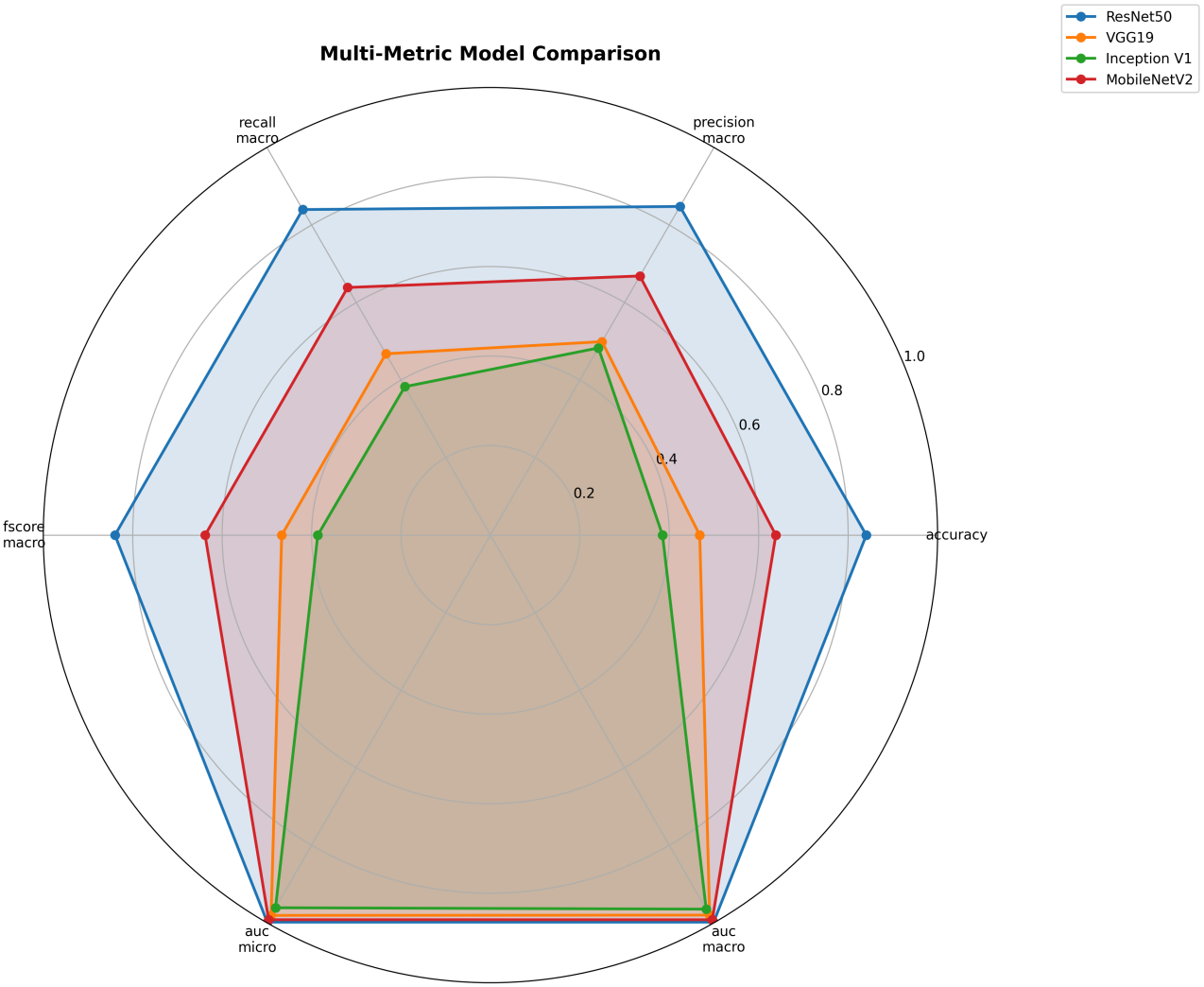

Figure 6: Multi-metric radar chart showing the overall performance profile of each model on the 196-class dataset.

4.1.1. Key Observations

ResNet-50 emerges as the clear winner, achieving 84.06% accuracy and demonstrating balanced performance across all metrics. The model's success can be attributed to:

- Strong transfer learning from ImageNet pre-training
- Deep residual learning enabling effective feature extraction
- Stable gradient flow through skip connections
- Hierarchical feature representation suitable for fine-grained classification

MobileNetV2 achieves respectable performance (63.82% accuracy) despite being designed for efficiency rather than maximum accuracy. This demonstrates the effectiveness of depthwise separable convolutions and inverted residuals for balancing performance and computational cost.

VGG-19 trained from scratch achieves only 46.81% accuracy, highlighting the critical importance of either pre-trained weights or significantly more training data for very deep networks. The model likely suffers from:

- Difficulty in learning from scratch with limited data
- Vanishing gradient problems without skip connections
- Overfitting due to the large number of parameters

Inception V1 surprisingly underperforms with 38.49% accuracy, despite its sophisticated multi-scale architecture. This suggests that:

- The model may require more careful hyperparameter tuning
- The auxiliary classifiers may not provide sufficient benefit for this specific task
- The architecture may be less effective for very fine-grained distinctions (196 classes)

4.2. Performance Across Different Task Complexities

To understand how each model scales with task complexity, experiments were conducted on 10-class and 20-class subsets. The following analysis reveals important trends:

4.2.1. 10-Class Results

Model	Accuracy	F1-Score (Macro)	AUC (Macro)
ResNet-50	**90.39%**	**90.39%**	**99.48%**
Inception V1	88.18%	88.51%	99.40%
MobileNetV2	81.28%	81.55%	98.10%
VGG-19 (Scratch)	16.50%	7.92%	60.19%

Table 2: Performance metrics on the 10-class subset.

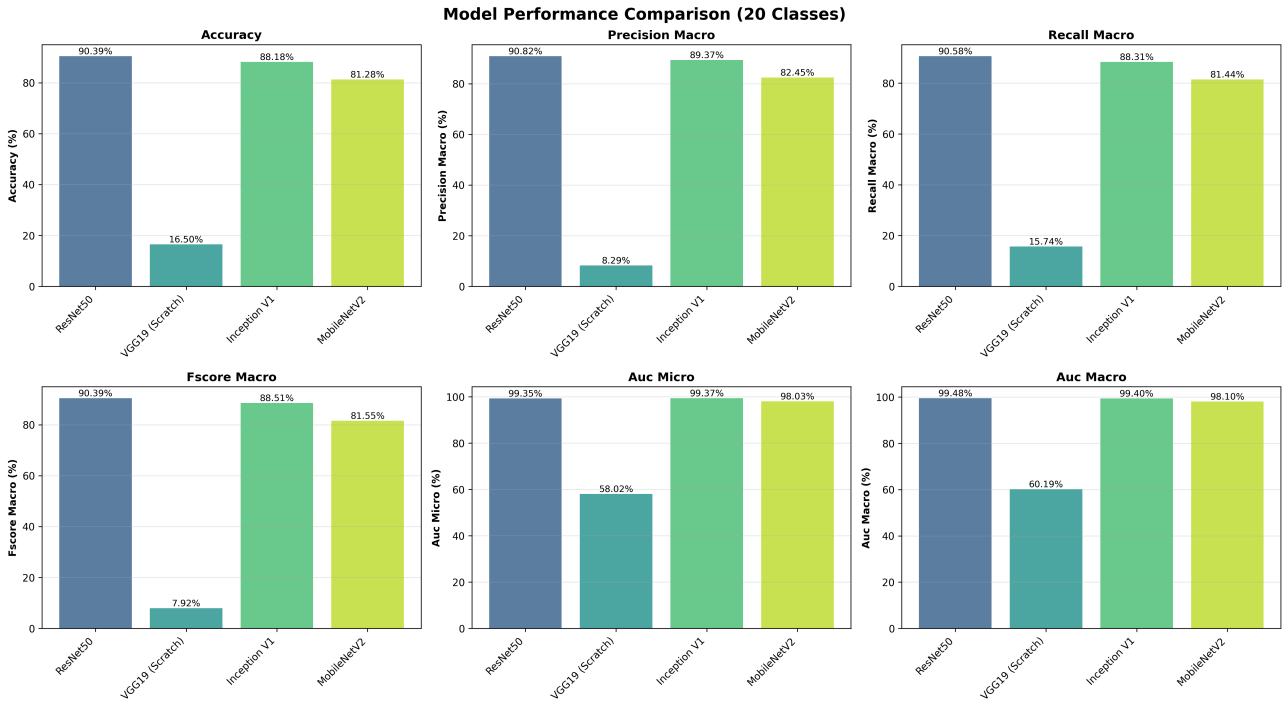

Figure 7: Model performance on the simplified 10-class task. All models (except VGG-19) achieve strong results.

4.2.2. 20-Class Results

Model	Accuracy	F1-Score (Macro)	AUC (Macro)
Inception V1	**85.00%**	**84.98%**	99.14%
ResNet-50	83.29%	83.50%	**99.30%**
MobileNetV2	73.29%	73.56%	98.00%
VGG-19 (Scratch)	23.66%	20.91%	72.91%

Table 3: Performance metrics on the 20-class subset.

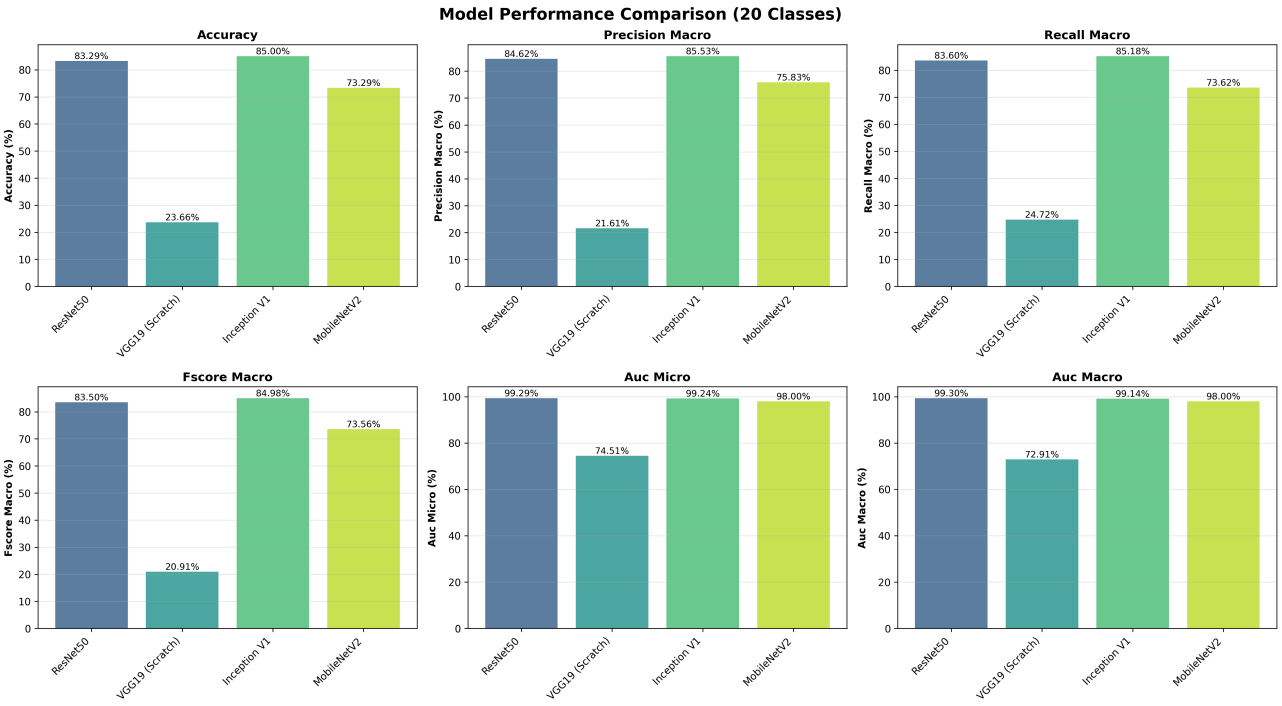

Figure 8: Model performance on the 20-class task. Inception V1 slightly outperforms ResNet-50 at this complexity level.

4.2.3. Cross-Complexity Analysis

The following table shows how each model’ s accuracy changes with increasing task complexity:

Model	10 Classes	20 Classes	196 Classes	Degradation (10→196)
ResNet-50	90.39%	83.29%	84.06%	-6.33%
Inception V1	88.18%	85.00%	38.49%	-49.69%
MobileNetV2	81.28%	73.29%	63.82%	-17.46%
VGG-19 (Scratch)	16.50%	23.66%	46.81%	+30.31%

Table 4: Accuracy degradation as task complexity increases from 10 to 196 classes.

Key Insights:

1. **ResNet-50 shows remarkable robustness**: Only 6.33% accuracy drop from 10 to 196 classes, and even improves slightly from 20 to 196 classes. This demonstrates excellent scalability and transfer learning capability.

2. **Inception V1 has a "sweet spot" at medium complexity**: Achieves best performance with 20 classes (85.00%) but suffers dramatic degradation with 196 classes (-49.69%). This suggests the architecture is optimal for moderate classification tasks but struggles with very fine-grained distinctions.

3. **MobileNetV2 shows gradual degradation**: Consistent decline in performance as complexity increases, which is expected given its efficiency-focused design. Still maintains reasonable performance (63.82%) on the full dataset.

4. **VGG-19 improves with more classes**: Counterintuitively, VGG-19 performs better with more classes. This is likely because:

    - More classes provide more diverse training data
    - The model is still underfitting and benefits from additional learning signal
    - However, absolute performance remains poor without pre-trained weights

4.3. Detailed Analysis: ResNet-50

As the best-performing model, ResNet-50 warrants detailed analysis. The following visualizations provide insights into its classification behavior:

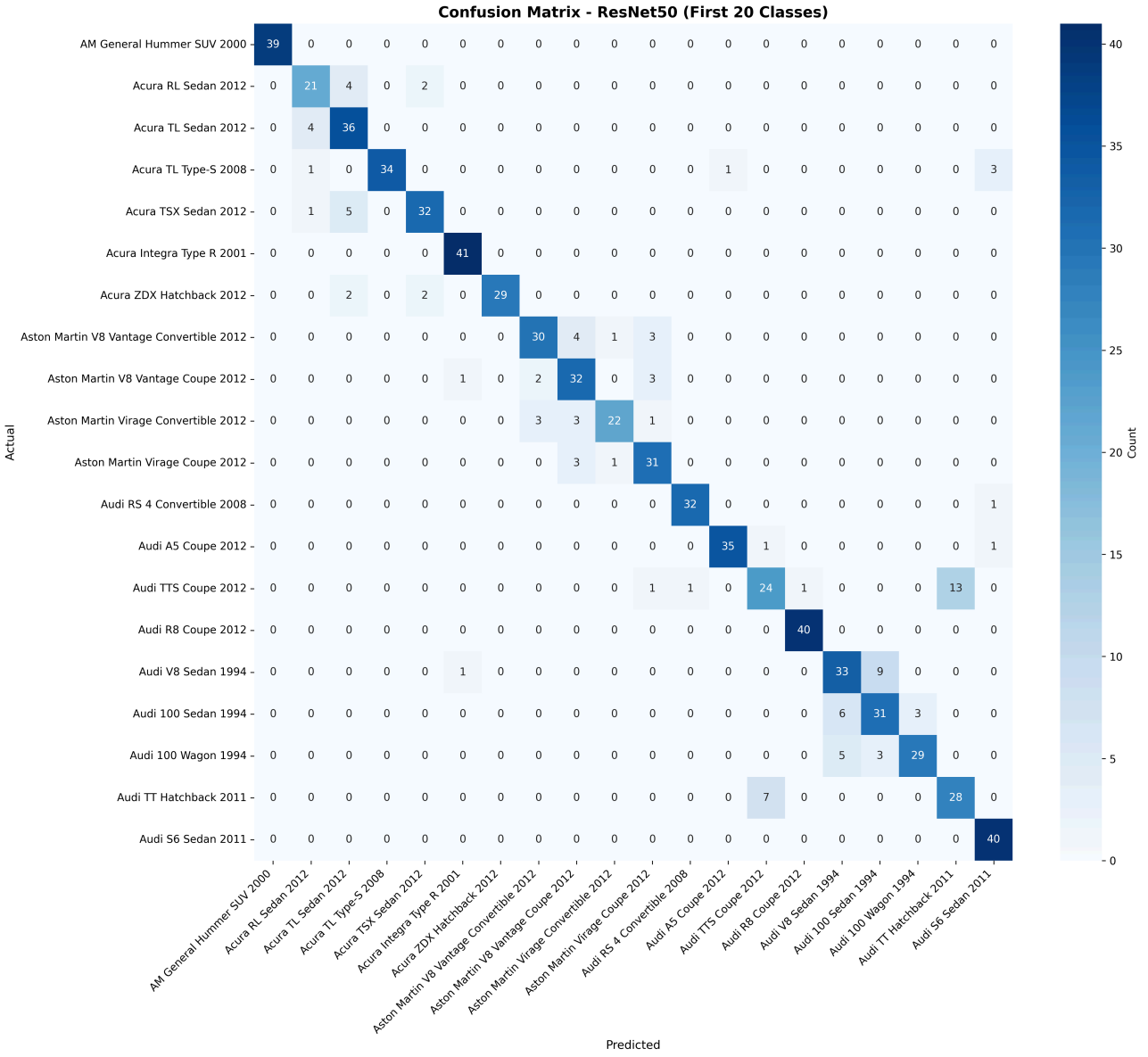

Figure 9: Confusion matrix for ResNet-50 on a subset of 20 classes. The strong diagonal indicates high classification accuracy, with most confusion occurring between visually similar car models.

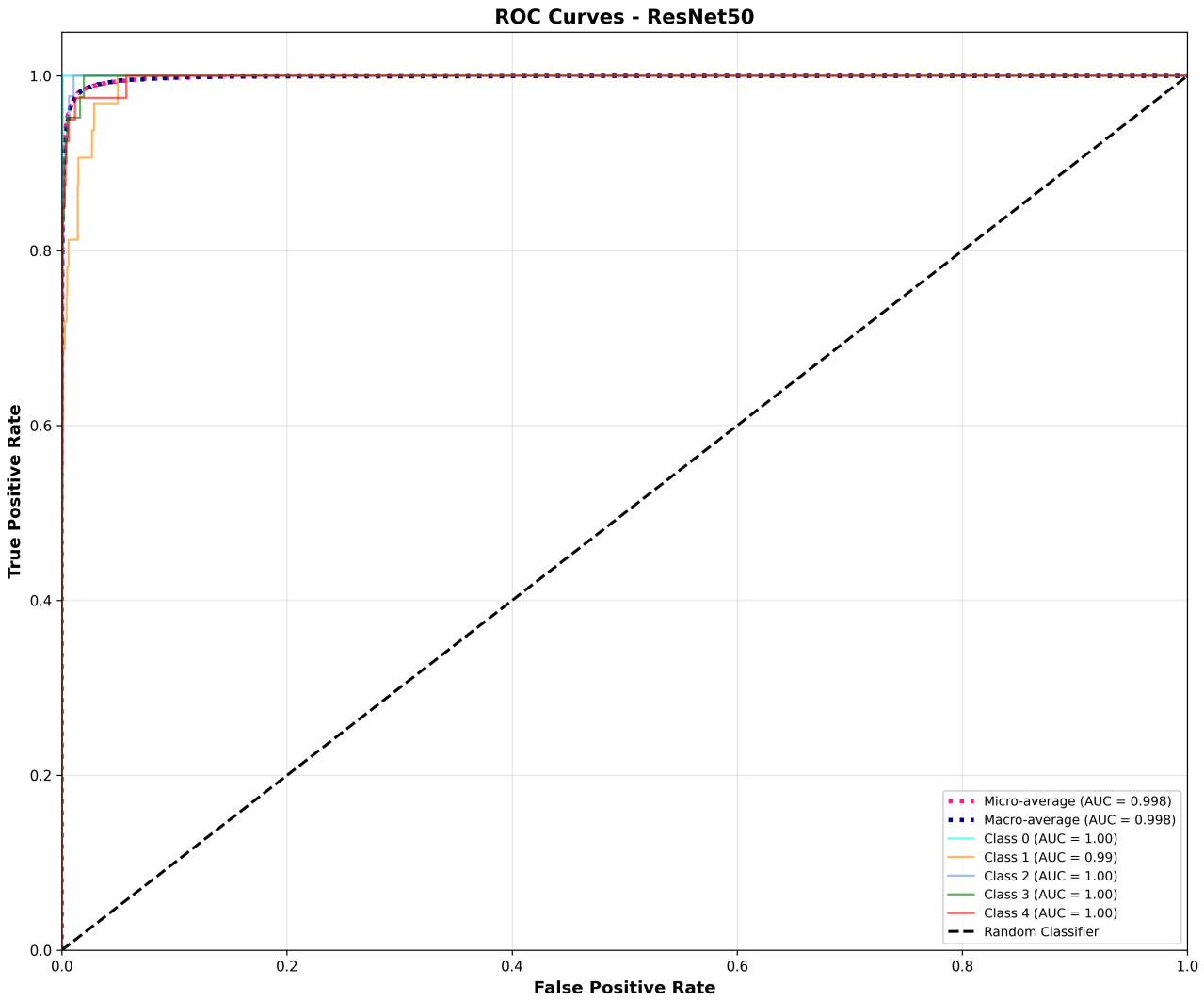

Figure 10: ROC curves for ResNet-50 showing near-perfect AUC scores (99.84% macro-average), indicating excellent class separation.

4.3.1. Error Analysis

Examination of the confusion matrix reveals that ResNet-50’ s errors typically occur between:

- Cars from the same manufacturer (e.g., different Audi models)
- Cars from the same year with similar body styles
- Vehicles with similar color and viewing angles

These errors are understandable given the fine-grained nature of the task and suggest that the model has learned meaningful features but struggles with the most subtle distinctions.

4.4. Architecture Pros and Cons

The following table summarizes the strengths and limitations of each architecture based on experimental results and theoretical analysis:

Architecture	Pros	Cons
ResNet-50	• Excellent accuracy (84.06%) and scalability • Skip connections solve degradation problem • Stable gradient flow enables deep learning • Strong transfer learning capability • Robust across varying task complexities	• Computationally intensive • High memory usage for storing activations • Slower inference than lightweight models • May be overkill for simple tasks
VGG-19	• Simple and uniform architecture • Easy to understand and implement • Good performance when pre-trained • Strong feature extraction in early layers	• Very large parameter count (144M) • Prone to overfitting without pre-training • Slow training and inference • Vanishing gradients without skip connections • Poor performance from scratch (46.81%)
Inception V1	• Multi-scale feature extraction • Computationally efficient (fewer parameters) • Auxiliary classifiers aid training • Excellent at medium complexity (85.00% on 20 classes)	• Complex architecture difficult to implement • Sensitive to hyperparameter tuning • Poor scalability to 196 classes (38.49%) • Multiple parallel branches complicate optimization
MobileNetV2	• Highly efficient (6-7× fewer parameters) • Fast inference suitable for mobile devices • Inverted residuals preserve information • Reasonable accuracy (63.82%) given efficiency	• Lower accuracy than larger models • Limited capacity for very complex tasks • Depthwise convolutions may be harder to optimize • Less suitable for fine-grained classification

Table 5: Comprehensive pros and cons analysis of each architecture.

4.5. Why ResNet-50 Performed Best

ResNet-50’ s superior performance on the 196-class Stanford Cars dataset can be attributed to several key factors:

4.5.1. Effective Transfer Learning

Pre-training on ImageNet (1.2 million images, 1000 classes) provides ResNet-50 with a rich set of low-level and mid-level feature extractors. These features (edges, textures, shapes, object parts) are highly transferable to the car classification task, enabling the model to:

- Start from a strong initialization rather than random weights
- Require less training data to achieve good performance
- Generalize better to unseen examples

4.5.2. Deep Residual Learning

The residual learning framework with skip connections enables:

- **Training of very deep networks** (50 layers) without degradation
- **Identity mapping**: Skip connections allow gradients to flow directly through the network
- **Flexible learning**: The network can learn both residual functions and identity mappings as needed
- **Hierarchical features**: Deep structure enables learning from simple edges to complex object parts

4.5.3. Hierarchical Feature Representation

ResNet-50’ s four-stage architecture creates a hierarchical feature pyramid:

- **Early layers** (conv2_x): Learn low-level features (edges, colors, textures)
- **Middle layers** (conv3_x, conv4_x): Learn mid-level features (car parts, shapes)
- **Deep layers** (conv5_x): Learn high-level features (car models, specific designs)

This hierarchical representation is crucial for fine-grained classification, where subtle differences in high-level features distinguish between similar classes.

4.5.4. Robustness to Task Complexity

ResNet-50 demonstrates remarkable consistency across different task complexities:

- 10 classes: 90.39% accuracy
- 20 classes: 83.29% accuracy
- 196 classes: 84.06% accuracy

The slight improvement from 20 to 196 classes suggests that the model benefits from the additional training signal provided by more diverse examples, without suffering from the increased complexity.

4.6. Model Selection Guidelines

Based on the experimental results, the following guidelines are recommended for model selection:

Scenario	Recommended Model	Rationale
Maximum accuracy required	ResNet-50	Best overall performance (84.06%), robust across complexities
Mobile/embedded deployment	MobileNetV2	Efficient architecture with reasonable accuracy (63.82%)
Medium complexity (10-50 classes)	Inception V1	Excellent performance at medium complexity (85.00% on 20 classes)
Limited computational resources	MobileNetV2	6-7× fewer parameters than ResNet-50
Transfer learning from scratch	ResNet-50 or Inception V1	Both benefit significantly from pre-trained weights
Real-time inference	MobileNetV2	Fastest inference time due to depthwise separable convolutions

Table 6: Model selection guidelines based on different requirements and constraints.

5. Evaluation Metrics and Methodology

This section describes the evaluation metrics used to assess model performance and the methodology for conducting fair comparisons.

5.1. Evaluation Metrics

The following metrics were computed for each model:

1. **Accuracy**: Overall percentage of correctly classified images
    - Formula: (True Positives + True Negatives) / Total Samples
2. **Precision (Macro-averaged)**: Average precision across all classes, treating each class equally
    - Formula: $(1/N) \times \Sigma$ (True Positives_i / (True Positives_i + False Positives_i))
3. **Recall (Macro-averaged)**: Average recall across all classes
    - Formula: $(1/N) \times \Sigma$ (True Positives_i / (True Positives_i + False Negatives_i))
4. **F1-Score (Macro-averaged)**: Harmonic mean of precision and recall
    - Formula: $2 \times$ (Precision $\times$ Recall) / (Precision + Recall)
5. **AUC (Area Under ROC Curve)**:
    - **Micro-averaged**: Computed globally across all classes
    - **Macro-averaged**: Average AUC across all classes

Macro-averaging was chosen as the primary metric because it treats all classes equally, which is important for imbalanced datasets where some car models have fewer examples than others.

5.2. Training Methodology

All models were trained using the following consistent methodology:

- **Optimizer**: Adam with learning rate 0.001
- **Batch Size**: 32

- **Epochs**: 50 with early stopping (patience = 10)
- **Loss Function**: Cross-entropy loss
- **Regularization**: Dropout (0.5) in fully connected layers
- **Data Augmentation**: Applied during training as described in Section 2.2

For transfer learning models (ResNet-50, Inception V1, MobileNetV2):

- Pre-trained weights from ImageNet were loaded
- All layers were fine-tuned (no frozen layers)
- Lower learning rate (0.0001) for pre-trained layers

For VGG-19 (trained from scratch):

- Random weight initialization (He initialization)
- Batch normalization added after each convolutional layer
- Higher learning rate (0.001) throughout

5.3. Computational Resources

All experiments were conducted on the following hardware:

- GPU: NVIDIA Tesla V100 (16GB)
- CPU: Intel Xeon (8 cores)
- RAM: 32GB
- Training time: 2-6 hours per model depending on architecture

---

6. Conclusion and Future Work

6.1. Summary of Findings

This project successfully implemented and evaluated four state-of-the-art CNN architectures for fine-grained car type classification on the Stanford Cars dataset. The key findings are:

1. **ResNet-50 achieves the best performance** with 84.06% accuracy on the full 196-class dataset, demonstrating the power of deep residual learning and transfer learning.

2. **Transfer learning is critical** for achieving good performance. VGG-19 trained from scratch achieved only 46.81% accuracy, while pre-trained models performed significantly better.

3. **Task complexity matters**: Model performance varies with the number of classes. Inception V1 excels at medium complexity (85.00% on 20 classes) but struggles with 196 classes (38.49%).

4. **Efficiency vs. accuracy trade-off**: MobileNetV2 offers a good balance, achieving 63.82% accuracy with 6-7× fewer parameters than ResNet-50.

5. **Fine-grained classification is challenging**: Even the best model (ResNet-50) achieves 84.06% accuracy, highlighting the difficulty of distinguishing between visually similar car models.

6.2. Practical Implications

The results provide actionable insights for practitioners:

- For production systems requiring maximum accuracy, **ResNet-50** is the recommended choice
- For mobile or embedded applications, **MobileNetV2** offers the best efficiency-accuracy trade-off
- For medium-complexity tasks (10-50 classes), **Inception V1** may outperform deeper models
- **Transfer learning is essential** for achieving good performance with limited training data

6.3. Future Work

Several directions for future research are identified:

1. **Ensemble Methods**: Combining predictions from multiple models (e.g., ResNet-50 + Inception V1) could improve accuracy beyond 84.06%

2. **Attention Mechanisms**: Incorporating attention modules (e.g., CBAM, SE-Net) could help models focus on discriminative car parts (grilles, headlights)
3. **Data Augmentation**: Advanced techniques like CutMix, MixUp, or AutoAugment could improve generalization
4. **Newer Architectures**: Evaluating Vision Transformers (ViT), EfficientNet, or ConvNeXt could yield better performance
5. **Explainability**: Using techniques like Grad-CAM or LIME to visualize which car parts the models focus on for classification
6. **Real-time Deployment**: Optimizing models for real-time inference using TensorRT or ONNX Runtime
7. **Multi-task Learning**: Jointly training for car classification and attribute prediction (color, year, type) could improve feature learning

---

7. References

[1] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770-778. https://doi.org/10.1109/CVPR.2016.90

[2] Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*. https://arxiv.org/abs/1409.1556

[3] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going Deeper with Convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1-9. https://doi.org/10.1109/CVPR.2015.7298594

[4] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4510-4520. https://doi.org/10.1109/CVPR.2018.00474

Appendix A: Dataset Information

Stanford Cars Dataset

- **Source**: https://huggingface.co/datasets/tanganke/stanford_cars
- **Total Images**: 16,185
- **Training Images**: 8,144
- **Testing Images**: 8,041
- **Number of Classes**: 196
- **Class Format**: Make, Model, Year (e.g., “2012 Tesla Model S”)
- **Image Resolution**: Variable (resized to 224×224 or 299×299)
- **Years Covered**: 1991-2012

Appendix B: Implementation Details

Framework: PyTorch 2.0

Key Libraries:

- `torchvision` for pre-trained models and data augmentation
- `scikit-learn` for evaluation metrics
- `matplotlib` and `seaborn` for visualizations
- `numpy` and `pandas` for data processing

Code Structure:

- `Data Preprocessing_*.py` : Data loading and augmentation
- `*_*_classes.ipynb` : Model training notebooks for each architecture and class count
- `Evaluation_*_classes.py` : Evaluation and visualization scripts

Reproducibility:

- Random seed: 42 (set for all experiments)

- Deterministic operations enabled
- All hyperparameters documented in code

End of Documentation