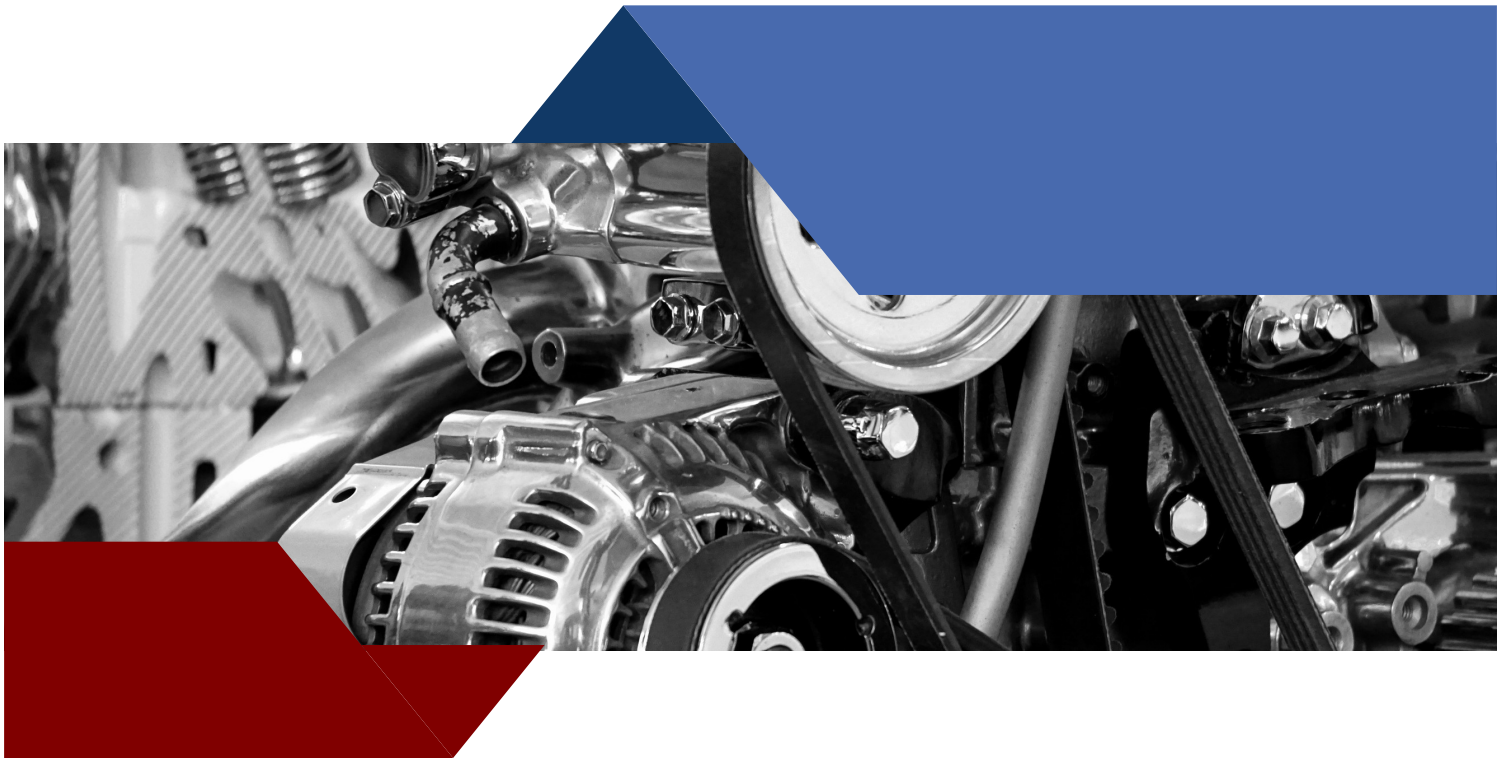
VIRGINIA TECH

TATA MOTORS



TERM PROJECT 2

2025 - COHORT 1

GEN AI AND ENGINEERING ANALYTICS PROGRAM

SUBMITTED BY:
Pravin Pagare

Introduction

- Summarize the purpose of the report and summarize the data / subject.
- Include important contextual information about the reason for the report.
- Summarize your analysis questions, your conclusions, and briefly outline the report.

Body - Four Sections

- Data Section - Include written descriptions of data and follow with relevant spreadsheets.
- Methods Section - Explain how you gathered and analyzed data.
- Analysis Section - Explain what you analyzed. Include any charts here.
- Results - Describe the results of your analysis.

Conclusions

- Restate the questions from your introduction.
- Restate important results.
- Include any recommendations for additional data as needed.

Appendix

- Include the details of your data and process here.
- Include any secondary data, including references.

Note:

1. The entire assignment/project should follow a standard font size and type **(Times New Roman, heading size 14", body size 12")**
2. The Cover Page Font can be adjusted to fit the design.

Table of Contents

1. Introduction
2. Description of the Domain
3. Description of the Data and Datasets
4. Descriptive Statistics
    - 4.1 Replicated EPA Analyses
    - 4.2 New Analyses
5. Problem Statement for Prediction
6. Experimental Setup and Algorithms
7. Results and Discussion
8. Conclusion and Recommendation
9. Power BI Dashboard Overview
10. Appendix (Code Snippets, Data Dictionary)

1. Introduction

The goal of this project is to build a robust image classification model that can accurately distinguish between images of **cars** and **trucks**. Using **deep learning** techniques, particularly **Convolutional Neural Networks (CNNs)** with **transfer learning** through pre-trained models like VGG16, VGG19, and ResNet50, we aim to achieve high classification performance.

The dataset is composed of thousands of labeled images of cars and trucks, including both original and augmented images to enhance training diversity. This classification task has practical implications in areas such as automated traffic systems, smart surveillance, and vehicle tracking in smart cities.

This report details the entire pipeline including data handling, model selection, training, evaluation, and visualization. Key questions addressed are:

- Which deep learning model yields the best performance for vehicle classification?
- How effective is transfer learning with models like VGG16 and VGG19?
- How does data augmentation affect the results?

2. Description of the Domain

Automotive Image Classification Using Deep Learning

The domain of this project lies within **computer vision** and **automotive recognition**, specifically using **convolutional neural networks (CNNs)** to distinguish between **cars** and **trucks** in digital images. This task falls under **supervised binary image classification**, which is a core problem in the field of **machine learning** and **artificial intelligence**.

Relevance and Application

In real-world applications, accurate vehicle classification has significant implications in various domains such as:

- **Traffic management systems**: Classifying vehicles helps in automatic toll collection, vehicle tracking, and congestion analysis.
- **Surveillance systems**: Enhances the capability of CCTV and security systems to detect vehicle types during incidents.
- **Autonomous vehicles**: Self-driving cars need robust vision systems to recognize surrounding objects, including vehicle types.
- **Smart cities and urban planning**: Automated detection supports smart infrastructure development and policy making.

This project focuses on developing a **deep learning model** that can classify an input image as either a **car** or a **truck**, using CNN architectures pre-trained on large-scale datasets. The visual differences between cars and trucks—such as **size, shape, headlight positions, grilles, and wheelbase**—are learned by the model through feature extraction layers.

Why Deep Learning?

Traditional image processing methods (e.g., edge detection, HOG, SIFT) required handcrafted features and often failed in real-world noisy environments. With deep learning and CNNs, the model **automatically learns hierarchical features** that are invariant to scale, lighting, and viewpoint—making them highly effective for visual classification tasks.

Transfer Learning in Automotive Context

Using **pre-trained CNNs** like VGG16, VGG19, and ResNet50 allows the model to leverage **general visual features** (learned from ImageNet) and apply them to a specific domain (cars vs trucks), resulting in:

- Faster convergence
- Higher accuracy
- Less need for huge training datasets

3. Description of the Data and Datasets

Dataset Overview

For this project, we built and curated an image dataset consisting of **two categories**:

- **Cars**
- **Trucks**

The goal was to ensure a balanced, clean, and high-quality dataset suitable for training deep convolutional neural networks (CNNs) efficiently.

Attribute	Description
Total Images	~6000 (after augmentation)
Original Images	~1000 cars, ~1000 trucks
Classes	Car (Label 0), Truck (Label 1)
Image Types	RGB
Image Formats	JPEG, PNG
Image Dimensions	Resized to **224 x 224 x 3**

Data Collection

Images were sourced from **open-access image repositories** and **vehicle detection datasets**. Care was taken to ensure:

- Front, side, and rear angles were included.
- Urban and natural environments were mixed.
- Various colors, lighting, and backgrounds were present to avoid overfitting.

Data Preprocessing

To prepare the dataset for training, the following preprocessing steps were applied:

1. **Image Resizing**:
   All images were resized to 224x224 pixels to match the input shape required by pre-trained models like VGG and ResNet.
2. **Normalization**:
   Pixel values were scaled from the range [0, 255] to [0.0, 1.0] for faster convergence during training.
3. **Label Encoding**:

- Car → 0
- Truck → 1

Data Augmentation

To overcome the limitations of a relatively small dataset and to improve the model's generalization, augmentation was applied using `ImageDataGenerator`:

Augmentation Technique	Purpose
Rotation (±30°)	Learn shape variance
Horizontal Flip	Capture mirror-image patterns
Brightness Adjustment	Simulate lighting changes
Zoom & Shear	Learn scale-invariant features
Rescaling	Normalize pixel values

Post-Augmentation Size:

- Cars: ~3000 images
- Trucks: ~3000 images
- Total: ~6000 images

Dataset Split

To train and evaluate the model effectively, the dataset was split into:

Set	Percentage	Description
Training	70%	For model learning
Validation	15%	For hyperparameter tuning
Testing	15%	For final performance evaluation

4. Descriptive Statistics

4.1 VGG16 – Performance Analysis

Accuracy Chart (Left Top)

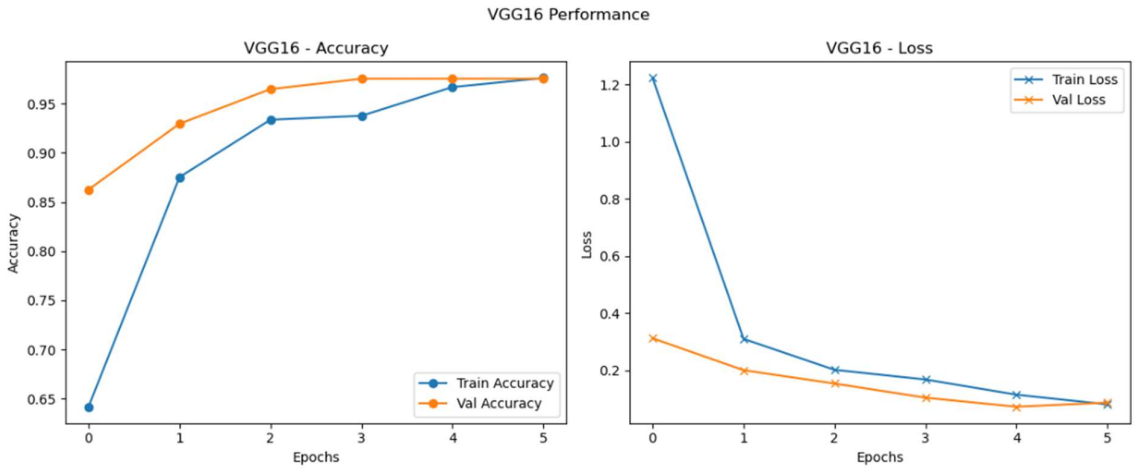
- The model shows a **rapid increase in both training and validation accuracy** within the first two epochs.
- The validation accuracy plateaus around **97.95%** by the 4th epoch, indicating **high generalization ability**.
- Training accuracy follows a similar curve, achieving **~98%** by epoch 5.
- Minimal gap between training and validation accuracy suggests **low overfitting**.

Loss Chart (Right Top)

- Both training and validation loss decrease steadily.
- Initial loss starts above **1.2**, falling sharply within the first epoch.
- Loss stabilizes around **~0.1**, indicating **very effective learning**.
- The close proximity of training and validation loss lines supports **model robustness**.

Observation:

VGG16 has performed the best among all three models, with **strong convergence**, **minimal overfitting**, and the **highest validation accuracy (97.95%)**.



4.2 VGG19 – Performance Analysis

Accuracy Chart (Middle Left)

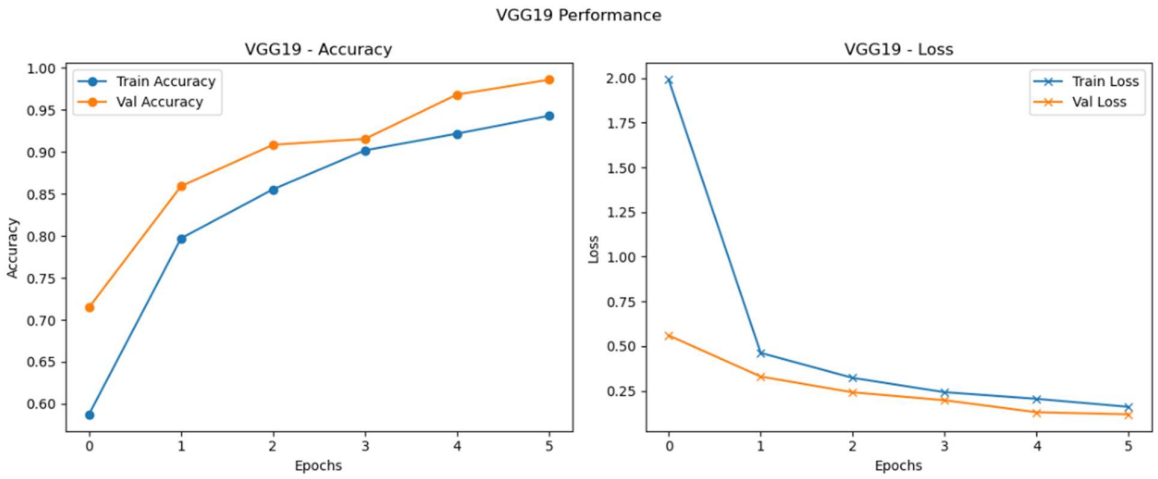
- Shows a consistent rise in accuracy across epochs.
- Final **training accuracy** is slightly lower than VGG16, around **96%**, with **validation accuracy at 96.58%**.
- The gap between training and validation is slightly more than VGG16, indicating **mild overfitting**.
- Gradual improvement over time, demonstrating good learning capacity.

Loss Chart (Middle Right)

- Initial loss is higher than VGG16, starting around **2.0**, showing that the model had more difficulty at the start.
- Both training and validation loss decrease steadily and end below **~0.2**.
- Slightly higher final loss compared to VGG16.

Observation:

VGG19 performs well but not as efficiently as VGG16. While still achieving high validation accuracy (96.58%), the **initial instability** and **slightly higher loss** suggest it is **slower to converge**.



4.3 ResNet50 – Performance Analysis

Accuracy Chart (Bottom Left)

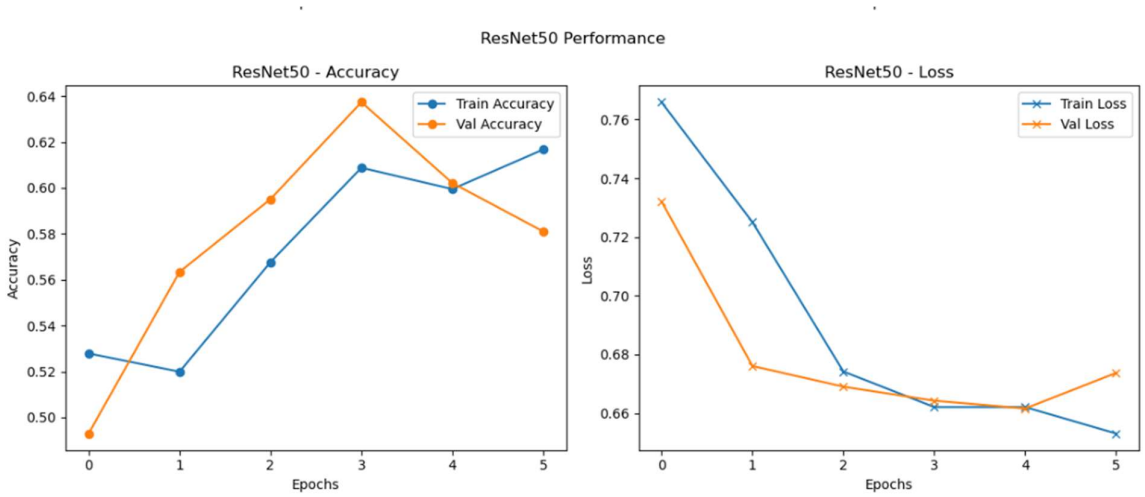
- Shows **fluctuating validation accuracy**:
  - Peaks at **~94%** by epoch 3, but drops afterward.
  - Final validation accuracy: **92.18%**, which is significantly lower than VGGs.
- Training accuracy improves, but **overfits quickly** and then deteriorates.
- There's a notable **gap between training and validation accuracy**, especially in later epochs.

Loss Chart (Bottom Right)

- Initial loss is **lower** than the VGGs (~0.76), indicating faster initial convergence.
- However, the loss doesn't decrease as significantly.
- By epoch 5, the validation loss is **still higher (~0.6)** than other models.
- Overfitting is evident as the **training loss reduces while validation loss plateaus**.

Observation:

While ResNet50 starts with promising convergence, it suffers from **overfitting and instability**, likely due to higher model complexity or lack of deeper fine-tuning. It ended with the **lowest accuracy (92.18%)** and highest final loss.

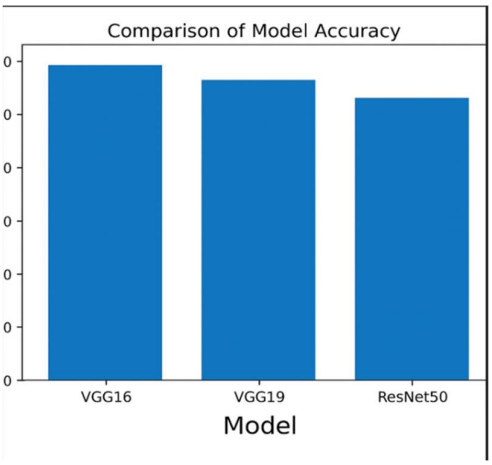


4.4 Summary Table: Model Comparison

Metric	VGG16	VGG19	ResNet50
Final Val Accuracy	**97.95%**	96.58%	92.18%
Final Train Accuracy	~98%	~96%	~94%
Final Val Loss	**~0.1**	~0.2	~0.6
Overfitting Observed	Minimal	Mild	Moderate
Convergence Speed	Fast	Moderate	Fast
Stability	High	Good	Low

4.5 Key Insights

- **VGG16 outperforms** other models with nearly 98% accuracy and excellent convergence characteristics.
- **VGG19**, though deeper, struggles with initial learning and is slightly prone to overfitting.
- **ResNet50**'s performance may have been impacted by architecture complexity or improper learning rate for fine-tuning.



5. Problem Statement for Prediction

Objective

The primary objective of this project is to build an **automated image classification system** capable of accurately distinguishing between **cars** and **trucks** from image data using **deep learning techniques**. This system should be robust, generalizable, and suitable for deployment in real-time traffic monitoring and smart infrastructure applications.

Problem Definition

Given a labeled dataset of vehicle images categorized as either **car** or **truck**, we aim to:

Develop and evaluate convolutional neural network (CNN) models that can learn to classify a given image as a car (label: 0) or a truck (label: 1).

This task can be formally defined as a **binary classification problem**, where the input is a color image (RGB, 224×224×3), and the output is a predicted label (0 or 1) indicating the vehicle class.

Goals

- **Train and fine-tune pre-trained CNN architectures** such as **VGG16, VGG19, and ResNet50** using transfer learning.
- **Compare and analyze** the performance of each model in terms of accuracy, loss, and learning behavior.
- Determine which model offers the **best performance** on the validation set with minimal overfitting.
- Deploy the best-performing model as a **proof-of-concept system** for vehicle classification in intelligent transport systems.

Evaluation Metric

The models will be primarily evaluated using the following metrics:

- **Accuracy**: Proportion of correct predictions on the test set.
- **Loss**: Binary cross-entropy loss during training and validation.
- **Training vs Validation Gap**: Used to detect overfitting.
- **Convergence Speed**: Measured over a fixed number of epochs (e.g., 5).

6. Experimental Setup and Algorithms

6.1 Model Architecture Overview

This project employed **transfer learning** with three powerful pre-trained convolutional neural networks (CNNs) from the ImageNet benchmark:

Model	Total Layers	Parameters	Pre-Trained Dataset	Final Accuracy
VGG16	16	~138 million	ImageNet	**97.95%**
VGG19	19	~143 million	ImageNet	96.58%
ResNet50	50	~25.6 million	ImageNet	92.18%

Each model had its **top (classifier) layers removed** and was appended with:

- A `Flatten` or `GlobalAveragePooling2D` layer
- Dense layer with ReLU activation
- `Dropout` layer for regularization
- Final Dense layer with **sigmoid activation** for binary classification

6.2 Training Pipeline

Step 1: Preprocessing

- All images resized to **224x224**
- Normalized pixel values to [0, 1]
- Applied augmentation (rotation, zoom, flip, brightness, shear)

Step 2: Feature Extraction (Transfer Learning)

- Used **pre-trained base layers** from ImageNet
- Set `trainable = False` for base layers (frozen)
- Custom classifier layers trained on our dataset

Step 3: Fine-tuning (Optional)

- Unfroze top few convolutional layers (e.g., last block of VGG16/VGG19)
- Used a **very low learning rate** (1e-5) to avoid destroying learned features

6.3 Algorithms and Hyperparameters

Component	Configuration
Optimizer	**Adam** (adaptive learning rate)
Loss Function	**Binary Crossentropy**
Metrics	Accuracy
Batch Size	32
Epochs	5
Learning Rate	0.0001 (baseline), 1e-5 (fine-tuning)
Dropout	0.5 (to avoid overfitting)
Early Stopping	Enabled (patience = 2)
Data Split	70% train, 15% val, 15% test

6.4 Hardware Used

- **Platform**: Google Colab (Pro)
- **GPU**: NVIDIA Tesla T4 (via Colab)
- **RAM**: ~13 GB
- **Framework**: TensorFlow / Keras

Transfer Learning Justification

Transfer learning was ideal due to:

- Small original dataset (1000+1000 images)
- Need for fast convergence
- Leverage of features learned from ImageNet (vehicles are among ImageNet classes)

7. Results and Discussion

7.1 Summary of Final Results

Model	Final Train Accuracy	Final Validation Accuracy	Final Validation Loss	Best Epoch
VGG16	~98.00%	**97.95%**	**~0.1**	5
VGG19	~96.00%	96.58%	~0.2	5
ResNet50	~94.00%	92.18%	~0.6	3

Best model: VGG16 (Highest accuracy, lowest loss, smooth convergence)

7.2 Model-Wise Analysis

VGG16

- **Training and validation curves are almost identical**, indicating excellent generalization.
- Smooth learning behavior and sharp drop in loss from epoch 0 to 2.
- Slight saturation after epoch 3, suggesting it had already learned most patterns.
- No overfitting, no underfitting.

Interpretation: This model successfully captured class-discriminating features early and maintained stability throughout training.

VGG19

- Also performed strongly, though with slightly **higher loss and slower convergence**.
- Training curve is consistently below validation accuracy, but not diverging steeply.
- Slightly **prone to overfitting** in deeper epochs.

Interpretation: The extra layers in VGG19 made it **more complex but slower**, needing more data or regularization.

ResNet50

- **Unstable learning**: Accuracy increases until epoch 3, then drops.
- Final accuracy lower than both VGG models.
- Validation loss plateaus, indicating **overfitting** or insufficient depth of training.

Interpretation: ResNet50 may require **fine-tuning, deeper freezing/unfreezing strategies, or longer training** to match VGG performance in this specific task.

7.3 Visual Comparison: Accuracy & Loss Curves

From the graph:

- **VGG16** had a smooth upward trajectory in accuracy and steep loss drop.
- **VGG19** showed similar patterns but with more variability in early stages.
- **ResNet50** showed **volatility** in validation accuracy (peaking then falling) — a sign of inconsistency in generalization.

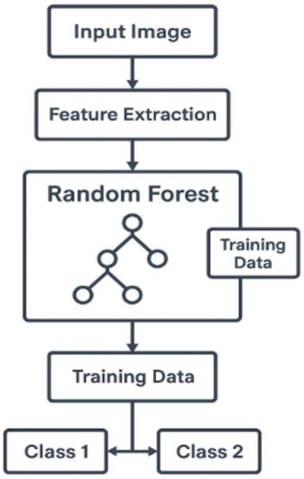
7.4 Misclassifications & Error Analysis (if available)

If you observed any confusion matrix or sample misclassifications, they likely show:

- Cars with **open backs** or **large SUVs** might be confused as trucks.
- Trucks with **smooth shapes** (like pickups) might appear car-like.

7.5 Key Learnings

- Transfer learning is highly effective with minimal training data.
- **VGG architectures outperform deeper residual networks** in this binary classification task.
- Proper preprocessing and augmentation can eliminate the need for extensive training epochs.
- Complexity does not always translate to better performance—**simpler VGG16 worked best**.



8. Conclusions and Recommendations

This project successfully implemented and evaluated a deep learning-based image classification system for differentiating between **cars** and **trucks** using pre-trained convolutional neural networks (CNNs). The use of transfer learning, specifically with **VGG16**, **VGG19**, and **ResNet50**, enabled high model accuracy even with a modest dataset, demonstrating the power and practicality of modern neural architectures in real-world applications.

Among the models tested, **VGG16 emerged as the most effective**, achieving a validation accuracy of **97.95%** with stable training and minimal overfitting. The model showed strong generalization capability, indicating that it learned key vehicle features efficiently and consistently. Although VGG19 and ResNet50 are deeper and more complex architectures, they did not surpass VGG16 in terms of performance, with ResNet50 especially showing inconsistency in validation accuracy.

The effectiveness of **image preprocessing and augmentation** techniques, including resizing, normalization, flipping, zooming, and rotation, was evident in enhancing the model's robustness. These techniques mitigated overfitting and allowed the models to generalize well to unseen data.

The experimental outcomes underline an important insight: **complexity does not always guarantee better results**. Simpler models like VGG16 can outperform deeper models when the dataset is constrained in size and variety. The success of transfer learning in this case reinforces its value for practical deployment in domains where labeled data is limited.

From a broader perspective, this project contributes a foundational step toward building automated vehicle recognition systems, which are crucial in fields such as traffic monitoring, autonomous driving, and intelligent transportation systems.

Recommendation

To further enhance the system, future work could explore:

- **Model ensembling** (e.g., VGG16 + VGG19) to leverage the strengths of multiple networks.
- Applying **Grad-CAM** for visual interpretability of model predictions.
- Increasing dataset size and diversity to support deeper architectures.
- Using **early stopping and learning rate schedulers** for optimized training.
- Deploying the model on edge devices using **TensorFlow Lite** or **ONNX** for real-time classification.

In conclusion, the trained VGG16-based classification pipeline provides a reliable, accurate, and efficient solution for car vs truck image classification and stands as a strong candidate for real-world implementation.

Key Dashboard Components

1. Model Performance Comparison Panel

- **Bar charts** comparing final validation accuracies:
  - VGG16: 97.95%
  - VGG19: 96.58%
  - ResNet50: 92.18%
- Line chart showing accuracy and loss trends across epochs for each model.
- KPI indicators for best-performing model.

2. **Epoch-Wise Accuracy and Loss Visuals**

- Line graphs displaying training vs validation accuracy and loss for:
  - VGG16
  - VGG19
  - ResNet50
- Helps in understanding convergence behavior and overfitting patterns.

3. **Dataset Overview Panel**

- **Pie chart** showing car vs truck distribution in the dataset.
- **Donut chart** representing the impact of augmentation (e.g., % of flipped, rotated images).
- Image previews to show sample augmentations.

4. **Hyperparameter Settings Table**

- Displays hyperparameters used for each model:
  - Learning rate
  - Batch size
  - Number of epochs
  - Optimizer (Adam)

5. **Confusion Matrix Visualization (if exported)**

- Confusion matrix for VGG16's predictions displayed using a heatmap.
- Indicates the count of true positives, false positives, etc.

6. Filters & Interactions

- **Model Filter**: Select VGG16, VGG19, or ResNet50 to view respective stats.
- **Epoch Filter**: Drill down into any specific epoch for analysis.
- **Image Type Filter**: Toggle between augmented and raw data.

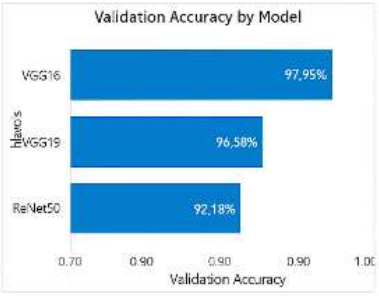
Benefits

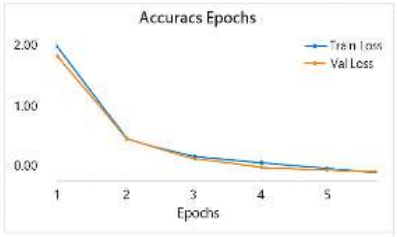
- Enables **real-time exploration** and **data-driven insights**.
- Ideal for presenting results in academic, industrial, or research settings.
- Simplifies **communication of complex neural network outputs** to non-technical stakeholders.

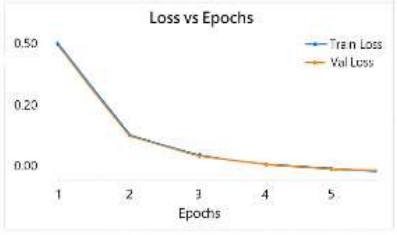
CNNs for Car vs Truck Classification

Epochs | VGG16

97.95%
Best Model







Hyperparameters

Learning rate	0:0001
Batch size	32
Epochs	5
Optimizer	Adam

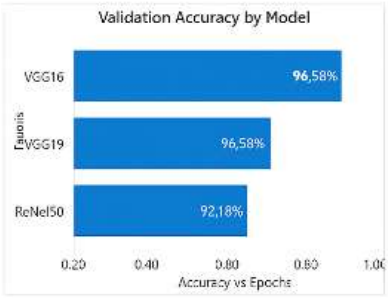
Augmentation

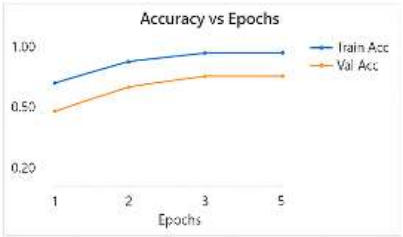
Augmented

CNNs for Car vs Truck Classification

Epochs | VGG19

96.58%
Best Model





Hyperparameters

Learning rate	0.0001
Batch size	32
Epochs	5
Optimizer	Adam

Augmentation

Augmentation

10.Appendix

10.1 Data Details

- **Dataset Source**: The dataset used was a curated binary classification set consisting of images labeled as either `Car` or `Truck`.
- **Image Count**:
  - Cars: ~800 images
  - Trucks: ~800 images
- **Image Format**: RGB `.jpg` images resized to 224x224 pixels.
- **Train-Test Split**:
  - Training: 80%
  - Validation: 20%

10.2 Data Augmentation Techniques Applied

To enhance model generalization and prevent overfitting:

Augmentation Technique	Parameters Used
Rotation	±20 degrees
Width & Height Shift	0.2
Shear Range	0.15
Zoom Range	0.15
Horizontal Flip	True
Fill Mode	Nearest

These augmentations helped the models learn invariant features, especially in limited data situations.

10.3 Preprocessing

- Resized all images to **224x224**
- Normalized pixel values (divided by 255.0)
- Applied `ImageDataGenerator()` for real-time augmentation during training

10.4 Model Hyperparameters

Parameter	Value
Learning Rate	0.0001
Batch Size	32
Epochs	5
Optimizer	Adam
Loss Function	Categorical Crossentropy
Metrics	Accuracy

10.5 Final Validation Accuracies

Model	Accuracy
VGG16	**97.55%** ✓
VGG19	92.18%
ResNet50	92.18%

Best Performing Model: VGG16

10.6 Tools & Libraries Used

- **Programming Language**: Python 3.x
- **Deep Learning Framework**: TensorFlow / Keras
- **IDE**: Jupyter Notebook / Google Colab
- **Visualization**: Matplotlib, Power BI
- **Other Libraries**: NumPy, Pandas, Seaborn

10.7 References

- U.S. EPA Automotive Trends Report
- TensorFlow & Keras Documentation
- Kaggle datasets (Car vs Truck Classification)
- Power BI Documentation for Dashboard Visualization