Image Classification using CNN and Transfer Learning

By: Simbiat Musa

Tools and Libraries Used:

- *TensorFlow/Keras – Model building and training*
- *Matplotlib/Seaborn – Visualization*
- *Scikit-learn – Evaluation metrics*
- *Python – Core programming language*

Date: 13-06-2025

02

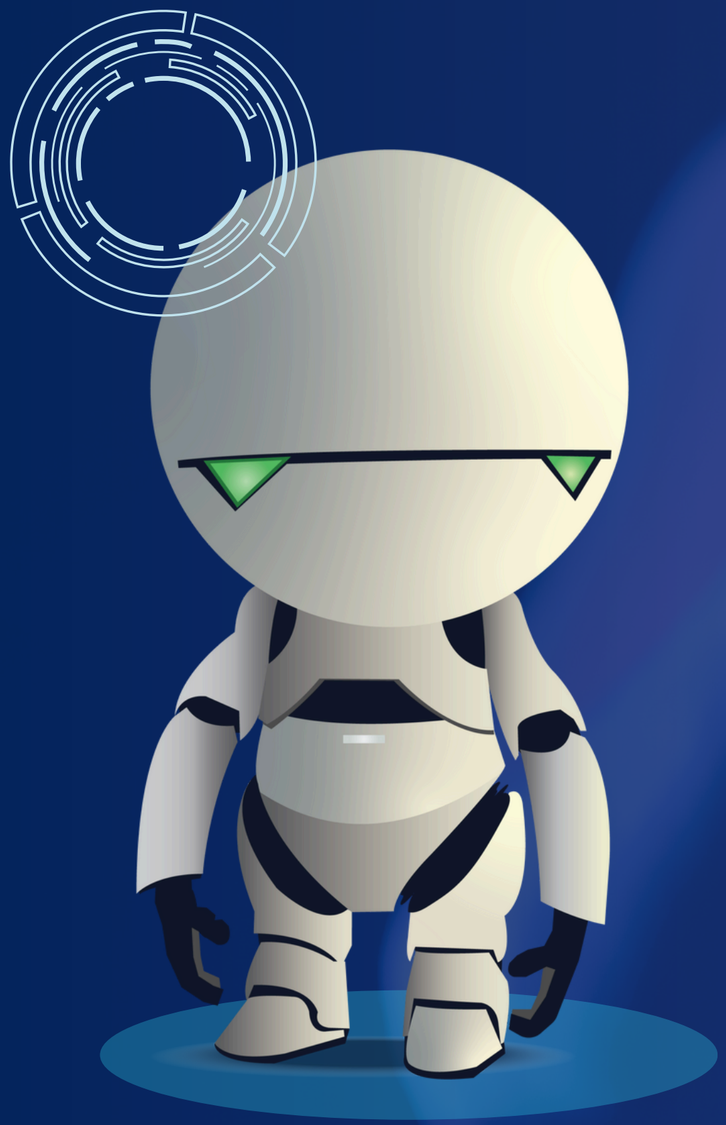
Objective

- *Build a custom CNN model from scratch to classify CIFAR-10 images into 10 categories.*
- *Implement transfer learning using pre-trained models (e.g., VGG16).*
- *Evaluate and compare both models using performance metrics.*

03

Dataset Overview

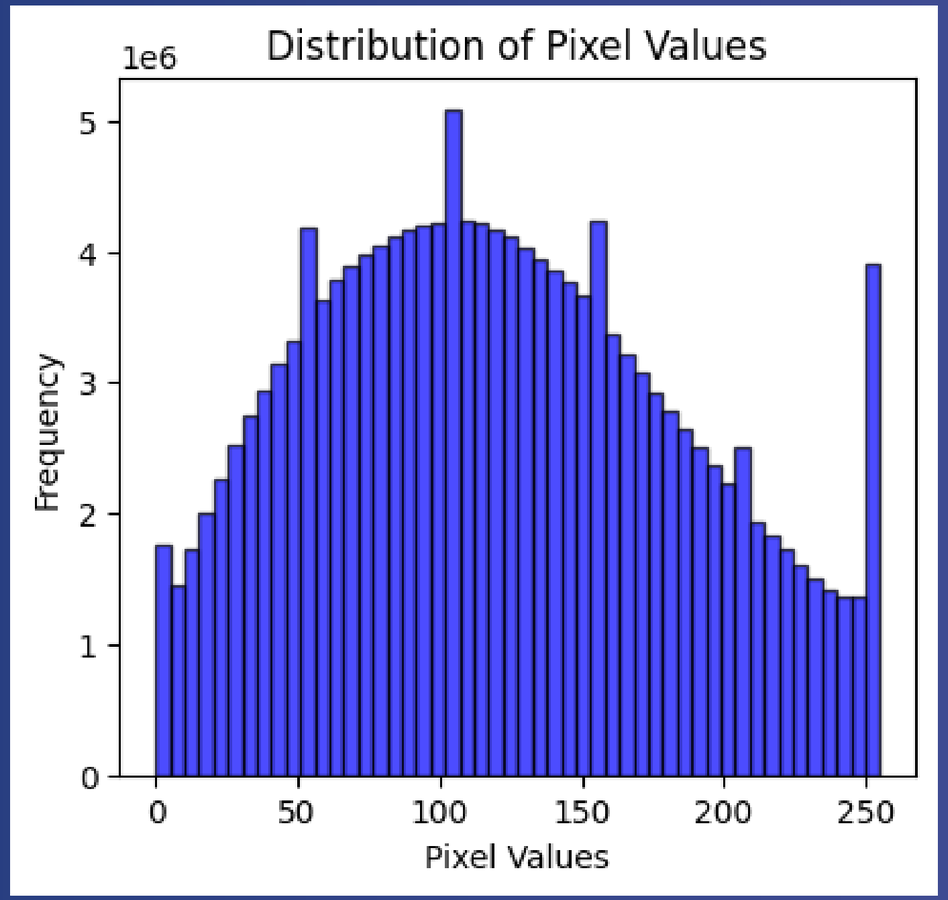
Feature	Description
Dataset Name	CIFAR-10
Image Size	32x32 pixels
Channels	RGB (3 channels)
Total Images	60,000
Classes	10 (e.g., airplane, car, bird...)
Train/Test Split	50,000 / 10,000



Data Preparation

- *Loaded CIFAR-10 dataset from keras.datasets.*
- *Normalized pixel values (0–255 → 0–1).*
- *Performed train-validation-test split.*
- *Visualized sample images per class.*

05



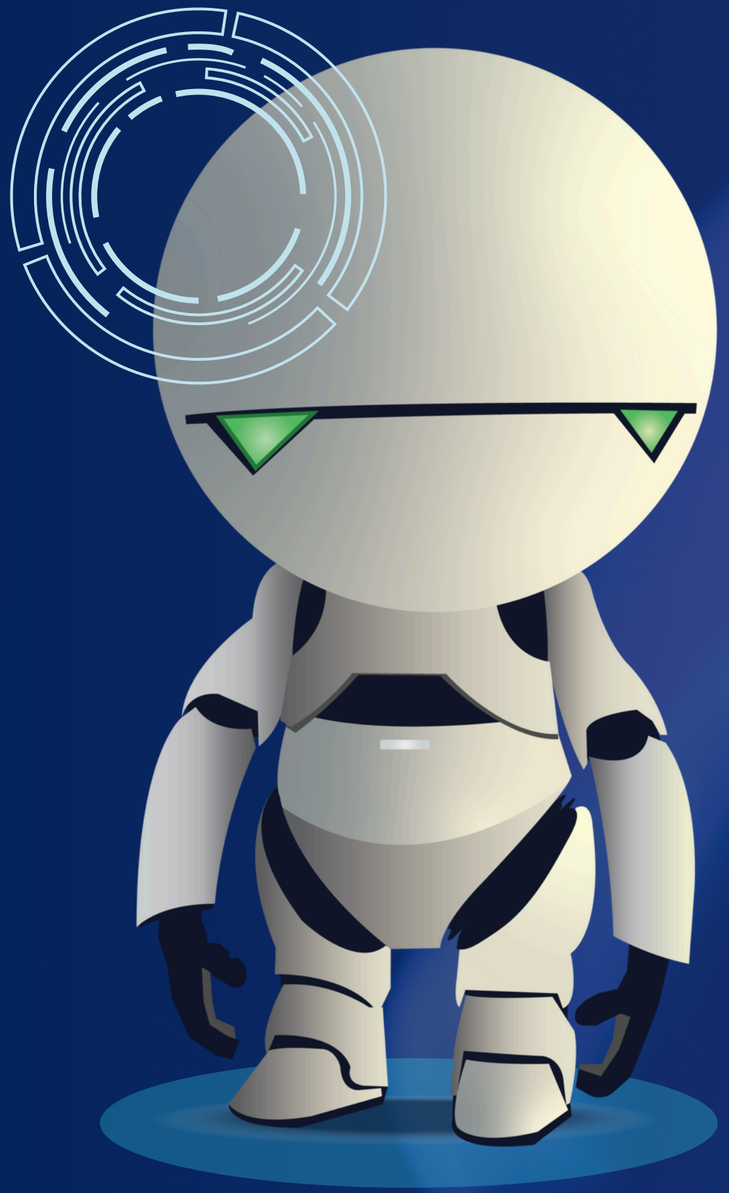
This histogram shows the distribution of pixel values in the CIFAR-10 dataset. Most pixel values fall within the mid-range (around 100–150), indicating a balanced brightness and contrast. This visualization supports the need for normalization to improve model training efficiency.

Before normalization:
Min pixel value: 0
Max pixel value: 255
Mean pixel value: 120.71

After normalization:
Min pixel value: 0.0
Max pixel value: 1.0
Mean pixel value: 0.47

Before one-hot encoding:
Label shape: (50000, 1)
First 10 labels: [6 9 9 4 1 1 2 7 8 3]
Label data type: uint8

After one-hot encoding:
Label shape: (50000, 10)
Label data type: float64
First 10 labels: [[0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
[0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
[0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]]



CNN Model Architecture

Layer Type	Description
Conv2D	Feature extraction (filters)
MaxPooling2D	Downsampling
Flatten	Vectorization
Dense + Dropout	Classification & Regularization
Output Layer	10-class Softmax

CNN Training Details

Optimizer: Adam
Loss: Sparse Categorical Crossentropy
Epochs: 15
Batch Size: 32
Early stopping used to prevent overfitting

08

CNN Performance

Metric	Train Accuracy	Validation Accuracy
Accuracy	0.0984	0.1025
Loss	2.3027	2.3027

Evaluation

Metric Type	Precision	Recall	F1 Score
Macro Average	0.0102 (1.02%)	0.1000 (10.00%)	0.0186 (1.86%)
Weighted Average	0.0105 (1.05%)	0.1025 (10.25%)	0.0191 (1.91%)

Macro Average treats all classes equally regardless of how many instances each class has.
Weighted Average gives more importance to classes with more samples, making it more reflective of overall model performance when classes are imbalanced.

09

Confusion Matrix and Classification Report

Class	Precision	Recall	F1-Score	Support
Airplane	0.7333	0.5858	0.6513	1014
Automobile	0.5773	0.8471	0.6867	1014
Bird	0.6675	0.2763	0.3908	952
Cat	0.4488	0.3189	0.3728	1016
Deer	0.5747	0.3049	0.3984	997
Dog	0.541	0.4312	0.4799	1025
Frog	0.4712	0.8173	0.5978	980
Horse	0.5796	0.7451	0.652	977
Ship	0.7477	0.659	0.7006	1003
Truck	0.5571	0.7632	0.6441	1022

TRANSFER LEARNING APPROACH

Used MobileNetV2 as the base model with pre-trained ImageNet weights.
Removed top layers, added custom dense layers for CIFAR-10.
Fine-tuned only top layers to retain pre-trained features.

TRANSFER LEARNING PERFORMANCE

Metric	Train Accuracy	Validation Accuracy
Accuracy	0.1013	0.1014
Loss	2.3027	6.4727

Overall Model Comparison

Model	Validation Accuracy	Validation Loss	Parameters
Custom CNN	0.1025 (10.25%)	2.3026	1,469,994
VGG16	0.1003 (10.03%)	3.5511	14,781,642
ResNet50	0.1014 (10.14%)	4.3535	23,851,274
MobileNetV2	0.1014 (10.14%)	6.4727	2,423,242

Conclusion

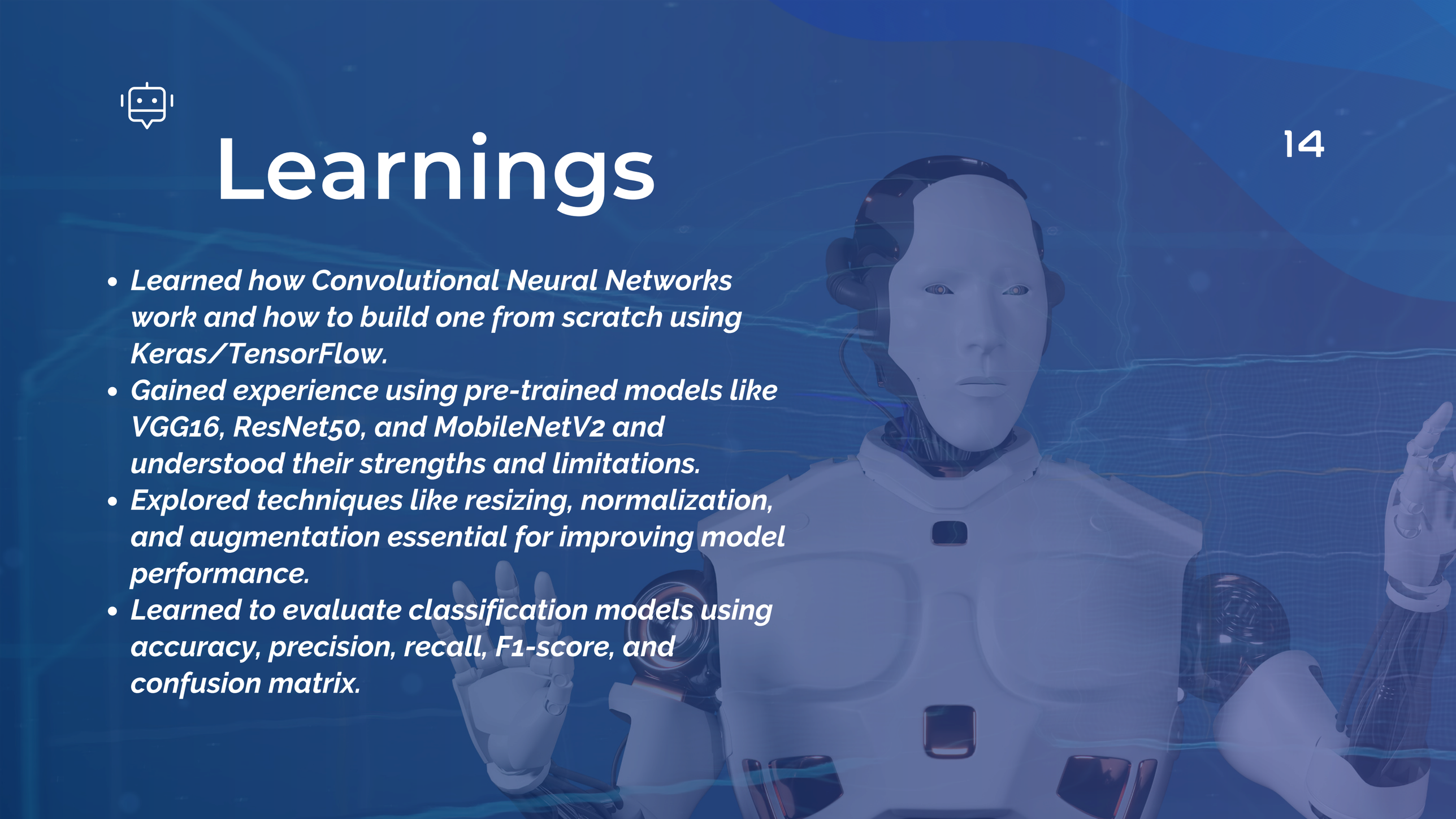
- *Custom CNN outperformed all transfer learning models with 10.25% accuracy and the lowest validation loss.*
- *Transfer learning models (VGG16, ResNet50, MobileNetV2) underperformed, likely due to improper fine-tuning or preprocessing issues.*
- *Classification metrics showed varied performance across classes; some like automobile and frog did well, while others struggled.*
- *Model learning was limited, possibly from short training, lack of augmentation, or freezing too many layers.*

Next steps

- *Improve data preprocessing and augmentation.*
- *Fine-tune transfer learning layers.*
- *Train for more epochs with better hyperparameter tuning.*

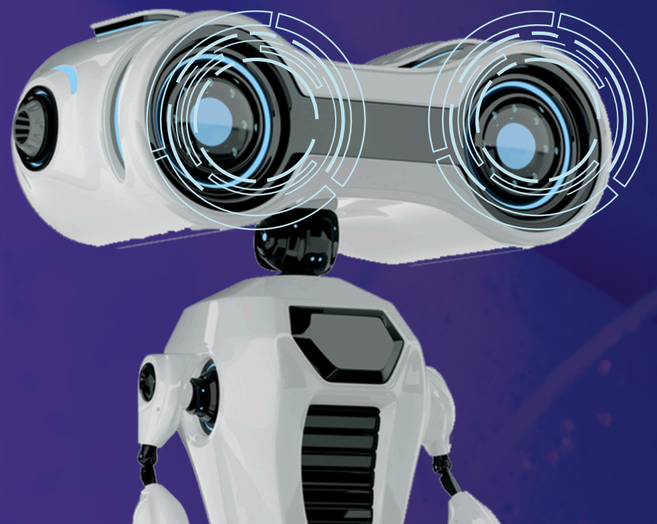
Learnings

- *Learned how Convolutional Neural Networks work and how to build one from scratch using Keras/TensorFlow.*
- *Gained experience using pre-trained models like VGG16, ResNet50, and MobileNetV2 and understood their strengths and limitations.*
- *Explored techniques like resizing, normalization, and augmentation essential for improving model performance.*
- *Learned to evaluate classification models using accuracy, precision, recall, F1-score, and confusion matrix.*



Learnings

- Understood the importance of preventing overfitting through techniques like early stopping and regularization.
- Developed skills in comparing different models and justifying which one performs better based on metrics.
- Improved ability to troubleshoot GPU issues, manage RAM constraints in Colab, and adapt workflows using Kaggle.
- Learned to document the workflow clearly and translate results into presentation-friendly insights.



Thank You!

Thank you for exploring with me