Deeploy CV Project

Assignment : 3

Name: Vaibhav Itauriya

Roll Number: 231115

ID : 223

Branch: ME

Date of submission: 26/12/2024

GitHub Repository of this Project:

---

1 Answer 1

Flag Detection: Differentiating Between the Flags of Poland and Indonesia. Flags of Poland and Indonesia share a similar design but differ in the placement of colors: Poland's flag has white on top and red at the bottom, whereas Indonesia's flag has red on top and white at the bottom. This similarity presents a challenging classification problem.

1.1 Initial Approach

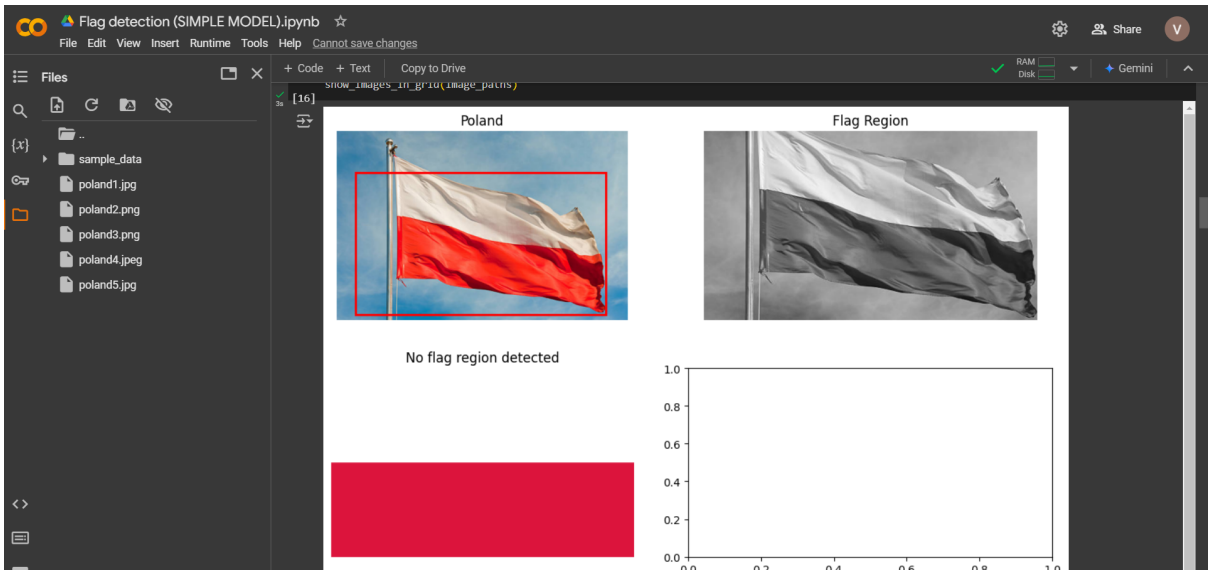

Figure 1: Intial Approach

Divide-by-Half Method:
This method involved dividing the image horizontally and analyzing each half. It failed to consider the precise locations of the colors, resulting in low accuracy.

Sobel Edge Detection:
Sobel operators were used to detect edges. However, this method was insufficient for

clear distinctions, as it lacked color-specific insights.

Pre-trained Models:
Open-source models like SSDVGG16 were explored. These models produced inconsistent results, as they were not trained specifically for flag datasets.

1.2 Optimized Methodology

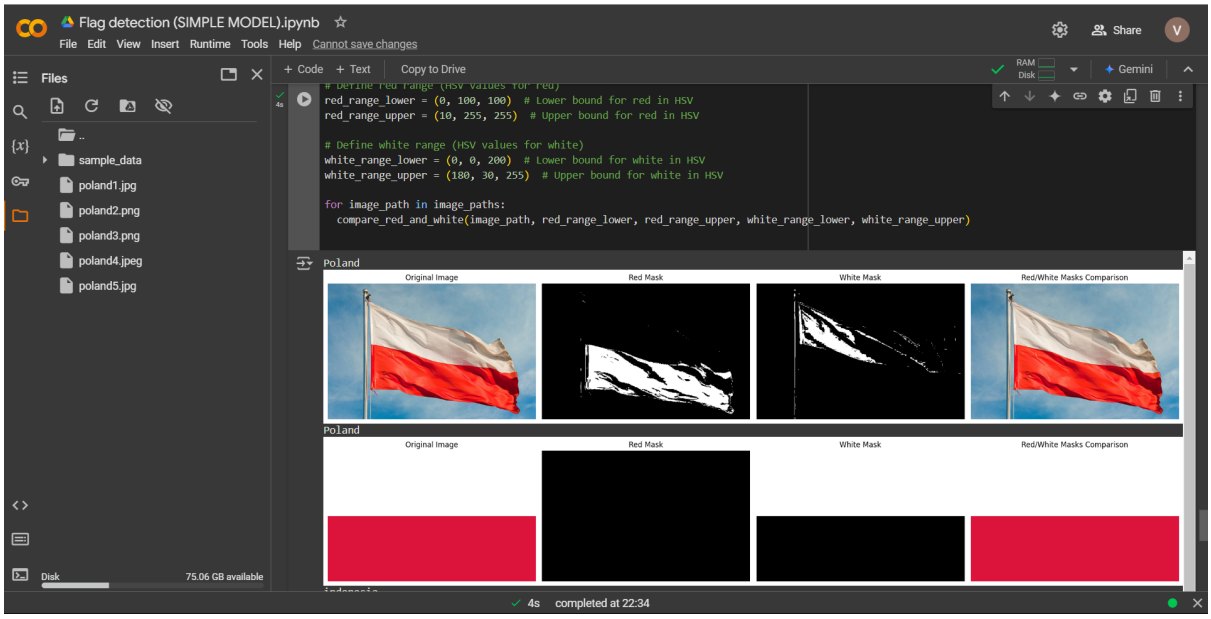

Figure 2: Optimized Methods

The optimized solution involved a combination of color masking in the HSV space and edge detection. The following steps were implemented:

HSV Color Masking:
Using HSV thresholds, red and white regions were isolated:

- Red and white masks were generated.
- Gaussian blur was applied to reduce noise.

Edge Detection:
Canny edge detection was used to identify contours of red and white regions.

Position Analysis:
The average vertical positions (y-coordinates) of red and white regions were compared:

- If the white region is above the red region, the flag is classified as **Poland**.
- Otherwise, it is classified as **Indonesia**.

1.3 Challenges and Improvements

Challenges:

- Noise in images led to misclassification.
- Similarity in flag designs required precise color thresholding.

Improvements:

- Gradient visualization was added to fine-tune HSV ranges.
- Smoothing and advanced edge-detection techniques improved accuracy.

1.4 Conclusion

The optimized approach using HSV and edge detection proved efficient for distinguishing between Poland and Indonesia flags. The combination of color-based segmentation and geometric position analysis ensured reliable detection.

---

2 Answer 2

2.1 Distance Measuring Modules

- **Euclidean:** Euclidean distance is the straight-line distance between two points in a Euclidean space. The Euclidean distance between two points p and q in n-dimensional space is given by:

  $$d(p, q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

  Use Case : *Used in clustering algorithms like K-Means, nearest neighbors, etc.*

- **Mahalanobis Distance:** Mahalanobis distance accounts for the correlation between variables and scales the data appropriately.

  The Mahalanobis distance is defined as:

  $$D_M = \sqrt{(x - \mu)^T W^{-1} (x - \mu)}$$

  where μ is the mean vector and W is the covariance matrix.

  Use case: *Outlier detection and classification.*

- **Manhattan Distance:** Also known as city block distance, it calculates the sum of absolute differences between the coordinates of two points:

  $$d(p, q) = \sum_{i=1}^{n} |q_i - p_i|$$

  Use case: *Used when movement is restricted to a grid (e.g., in pathfinding algorithms).*

2.2 Adam Optimizer

Adaptive Moment Estimation (Adam) adjusts the learning rate for each parameter. It uses moving averages of both gradients and their squares. The Adam optimizer updates parameters using:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

2.3 Loss Functions

- **L2 Regularized Loss:**
  The L2-regularized loss is given by:

$$L = \text{MSE} + \lambda \|W\|_2^2$$

- **Ridge Regularization:**
  Ridge regularization, is a form of L2 regularization, and is defined as:

$$\text{Loss} = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + \alpha \|W\|_2^2$$

2.4 Cross-Entropy Loss:

- **Binary Cross-Entropy Loss:**

$$L = -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \right]$$

- **Categorical Cross-Entropy Loss:**

$$L = -\sum_{i=1}^{N} y_i \log(p_i)$$

2.5 Activation Functions

- **Softmax:** Converts logic to probabilities, the Softmax activation function is:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

  Use case: *Multi-class classification.*

- **Sigmoid:** The Sigmoid activation function maps values to $(0, 1)$:

$$\sigma(x) = \frac{1}{1 + e^{-z}}$$

  Use case: *Binary classification.*

- **Tanh:** The Tanh activation function maps values to $(-1, 1)$:

$$\text{Tanh}(x) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

  Use case: *Hidden layers in neural networks.*

2.6 Learning Rate

Defines how quickly a model updates its weights during training. The learning rate η determines the size of each step toward the minimum of the loss function.

Use case: *Too high a learning rate can overshoot minima, and too low can slow convergence.*

2.7 Batches

Data is divided into smaller chunks (batches) for processing. Batch size determines how many samples are passed through the model in one go.

Use case: *Reduces memory requirements and speeds up training.*

2.8 Gradient Descent

- **Basic Gradient Descent:** The gradient descent algorithm updates parameters as:

$$\theta = \theta - \eta \nabla J(\theta)$$

- **Batch Gradient Descent:** Gradient is computed using the entire dataset.

- **Stochastic Gradient Descent:** Gradient is computed using a single random data point.

- **Mini-Batch Gradient Descent:** Gradient is computed using small batches of data points.

2.9 Explanation of Table

This table outlines how to decide the **loss function** and the **activation function** for the last layer in a neural network based on the classification task:

- **1 or 2 classes**:
  - **Class mode:** Binary classification involves two possible outcomes (e.g., Yes/No, True/False).
  - **Loss function:** `binary_crossentropy` measures the difference between predicted and actual binary labels, making it suitable for binary classification.

  - **Activation:** The `sigmoid` function outputs probabilities in the range [0, 1], ideal for binary tasks.

- **Multiclass, single label**:
  - **Class mode:** For problems with multiple classes, where each sample belongs to only one class.
  - **Loss function:** `categorical_crossentropy` evaluates the difference between predicted probabilities and true one-hot encoded labels.
  - **Activation:** The `softmax` function ensures output probabilities sum to 1, meeting the requirements for multiclass single-label classification.

- **Multiclass, multilabel**:
  - **Class mode:** Multilabel problems allow samples to belong to multiple classes simultaneously.
  - **Loss function:** `binary_crossentropy` treats each class as a separate binary task, allowing for multiple labels.
  - **Activation:** The `sigmoid` function outputs independent probabilities for each class, suitable for multilabel tasks.

2.10 Logistics Regression Model

The table below represents the dataset for predicting house prices (y) based on x_1 (size in square feet) and x_2 (number of bedrooms):

x_1 (Size in sqft)	x_2 (Bedrooms)	y (Price in $1000s)
1200	3	220
1500	4	280
1700	3	300
2000	5	400
2500	4	450
3000	5	520
3500	6	580
4000	6	640

Data Points

$$\boxed{y = Model : Price = 25.1515 + 0.1289 Sizeinsqfts + 19.1164 Bedrooms}$$

Our Model

Definitions of Evaluation Metrics

1. R-squared (R^2)

R-squared measures the proportion of variance in the dependent variable (y) explained by the independent variables (x_1, x_2).

Summary of Overall Fit

R-Squared:	r^2 =0.9863
Adjusted R-Squared:	r^2_{adj} =0.9808
Residual Standard Error:	20.8959 on 5 degrees of freedom.
Overall F-statistic:	179.7867 on 2 and 5 degrees of freedom.
Overall p-value:	0

Analysis of Variance Table

Source	df	SS	MS	F-statistic	p-value
Regression	2	157004.2976	78502.1488	179.7867	0
Residual Error	5	2183.2024	436.6405		
Total	7	159187.5	22741.0714		

Figure 3: Output

$$R^2 = 1 - \frac{\text{SS}_{\text{res}}}{\text{SS}_{\text{tot}}}$$

Where:

- SS_{res}: Sum of squared residuals.
- SS_{tot}: Total sum of squares.

2. Adjusted R-squared (R^2_{adj})

Adjusted R-squared adjusts R^2 for the number of predictors in the model:

$$R^2_{\text{adj}} = 1 - \left(\frac{(1 - R^2)(n - 1)}{n - k - 1} \right)$$

Where:

- n: Number of observations.
- k: Number of predictors.

3. p-value (p-statistic)

The p-value tests whether the coefficients of predictors are statistically significant. A small p-value (e.g., < 0.05) indicates strong evidence against the null hypothesis (H_0).

4. Standard Error (SE)

Standard error measures the accuracy of the estimated coefficients. Smaller SE indicates more reliable estimates.

5. F-statistic

The F-statistic tests the null hypothesis that all coefficients are zero:

$$H_0 : \beta_1 = \beta_2 = \ldots = \beta_k = 0$$

6. Regression Equation

The regression equation predicts y as a function of x_1 and x_2:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

2.11 Basis Function (ϕ)

The basis function transforms the input features into a space suitable for regression. For linear regression:

$$\phi(x) = [1, x_1, x_2]$$

This includes:

- 1: Intercept term.
- x_1: Feature 1 (size in sqft).
- x_2: Feature 2 (number of bedrooms).

---

3 Answer 3

Based on the given book, a **remarkable fact** I found: In *1995*, **NavLab 5** became the first autonomous vehicle to obtain a driver's license, completing an impressive **50 self-driven miles** within a *2,850-mile journey.* This pioneering achievement, which preceded Tesla by more than a decade, remains an often overlooked milestone in the evolution of self-driving technology.

---

4 Answer 4

Yes, this code intensifies the dark parts of the image, highlighting the areas the model focuses on, much like how night vision or thermal imaging goggles work. It helps make important features in the image more visible by "lighting them up."

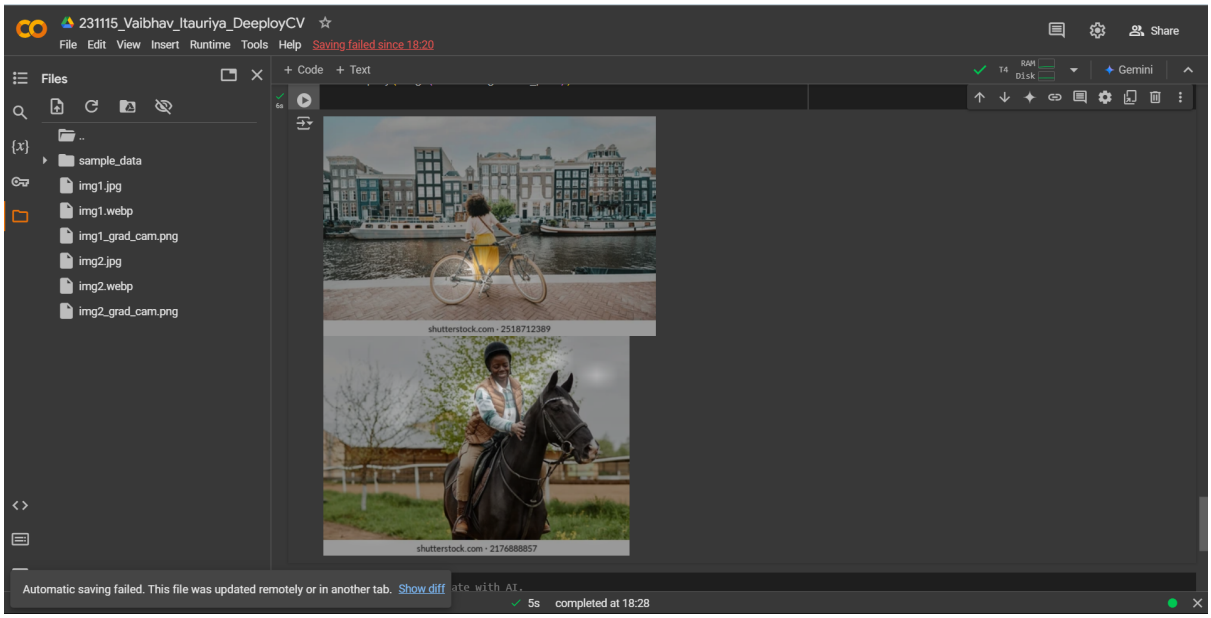

Figure 4: Output of Code

4.1 How it detects it?

This code uses Grad CAM (Gradient Weighted Class Activation Mapping) to generate a heat map that highlights areas of an image that contribute the most to the prediction of a neural network. The heatmap is overlaid on the original image to show which parts the model "attended to" when making its prediction, much like how thermal goggles highlight important areas in low-visibility environments.

This code essentially acts like thermal goggles for a neural network. It visually shows you which areas of the image are most important for the model's prediction. By intensifying these areas and darkening the less relevant ones, the code provides a clear visualization of the model's "focus," making it easier to understand its decision-making process.

4.2 Code Explanation

- **Preprocess Image:** The code begins by loading an image and resizing it to 224x224 pixels. This is necessary because the VGG16 model, which is pre-trained on ImageNet, expects input images of this size. The image is then converted into an array and preprocessed to match the format the model was trained on.
- **Grad-CAM Calculation:** In the generate_gradcam function, the code calculates the Grad-CAM heatmap. This is done by:
  - Using a gradient tape to calculate the gradients of the predicted class (which indicates the target object) with respect to the output of the last convolutional layer.
  - The gradients are averaged (pooled) across the spatial dimensions of the feature maps.
  - These averaged gradients are then multiplied by the corresponding feature maps to generate a heatmap that shows the areas in the image the model

focuses on.

- **Overlay Heatmap:** The generated heatmap is resized to match the original image size and blended with the original image. The intensity of the important regions (highlighted by the heatmap) is increased, while less important regions are darkened. This is what makes the important features "pop" in the final output, similar to how thermal goggles highlight heat.

  Link to Code: Code Link

---

5 Answer 5

I have done the implementation and observations of image classification using pre-trained deep learning models. The models were tested with generic images corresponding to predefined classes to evaluate their performance without bias or reliance on prior visual memories.

Setup and Methodology

IDE Selection:
Google Colab was used as the IDE for this implementation due to its computational resources and GPU support.

Image Selection:
The following generic images were chosen:

- **Apple**: A high-quality image of a red apple.
- **Bridge**: An image of a suspension bridge.
- **Wolf**: An image of a gray wolf in its natural habitat.

These images were selected carefully to ensure compatibility with the ImageNet classes without relying on pre-learned visual cues.

Models Used:
The following pre-trained models were evaluated:

- ResNet50
- ResNet152
- MobileNetV2
- InceptionV3

Implementation:
Each model was loaded with pre-trained ImageNet weights. Images were preprocessed to match the input size and format expected by each model, and predictions were decoded to display the top three classes for comparison.

Observations

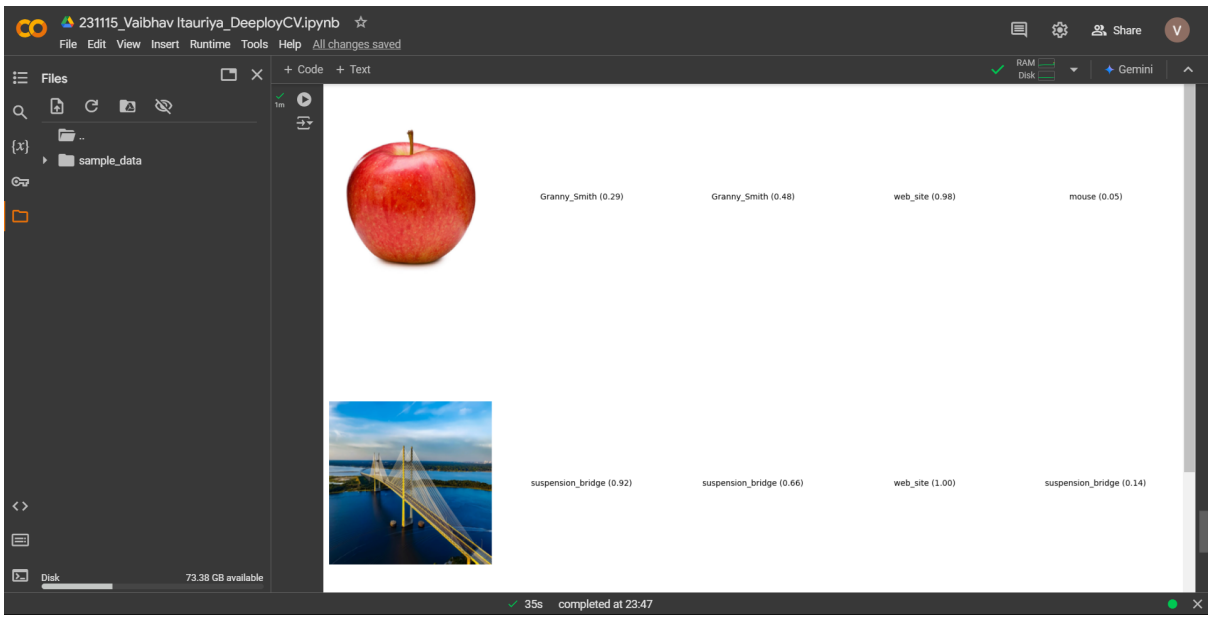

Figure 5: Top Output of: ResNet 50, ResNet152, MobileNetV2, InceptionV3

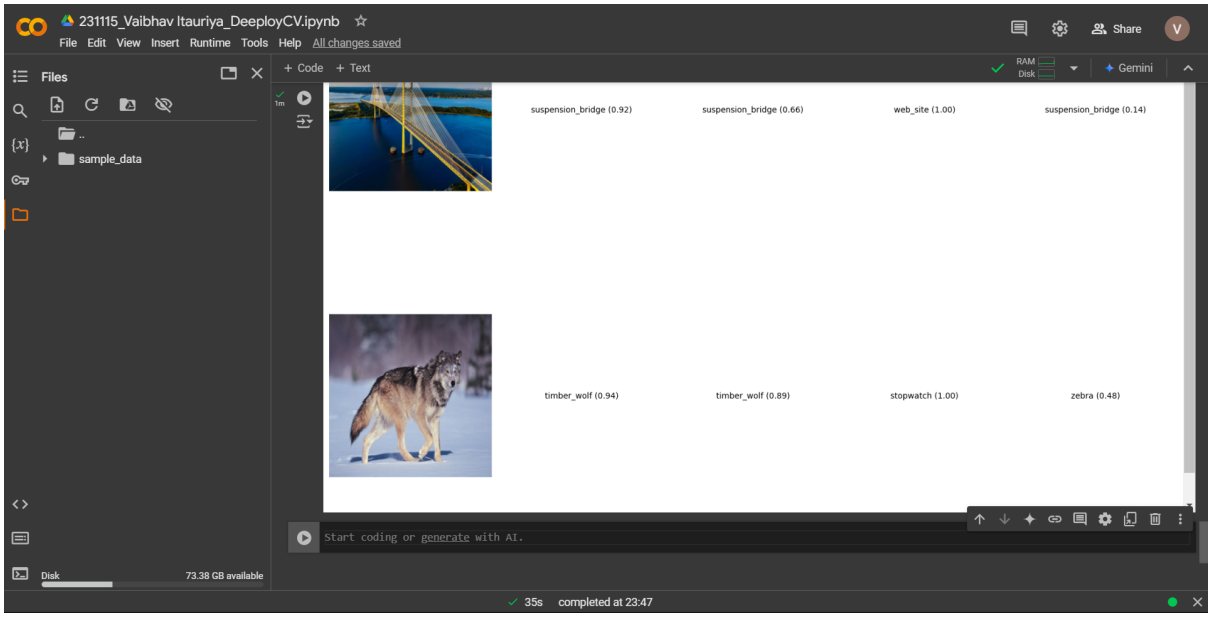

Figure 6: Top Output of: ResNet 50, ResNet152, MobileNetV2, InceptionV3

ResNet Models

- **ResNet50** and **ResNet152** exhibited exceptional accuracy, identifying not only the objects but also their specific subcategories (e.g., type of apple, type of bridge).

- **ResNet152** demonstrated higher confidence and provided more precise probabilities than ResNet50, benefiting from its deeper architecture.

MobileNetV2

- MobileNetV2 achieved good overall identification but lacked the accuracy of ResNet models.
- It struggled with detailed subcategories and provided less precise predictions for complex images.

InceptionV3

- InceptionV3 showed average performance, correctly identifying main objects but often misclassifying subcategories.
- It failed on certain images where ResNet models succeeded.

Conclusions and Insights

- **ResNet Models**: Best suited for tasks requiring high accuracy and detailed classification. ResNet152, in particular, is highly reliable.
- **MobileNetV2**: A lightweight alternative, suitable for scenarios with limited computational resources, but at the cost of reduced accuracy.
- **General Observations**: Deeper models like ResNet152 outperform others in capturing finer details, while lightweight models prioritize efficiency.
- **Practical Applications**: Model selection depends on the task requirements, such as speed, accuracy, or resource constraints.

Code Link: Click Here