Project Report: Real-Time Object Detection

Introduction

This project explores the application of two pre-trained deep learning models, VGG16 and MobileNetV2, for the purpose of real-time object detection using a webcam. The primary goal is to leverage these models to identify various objects in a live video stream, categorizing them into predefined classes such as 'brown-glass', 'cardboard', 'clothes', etc.

Model Descriptions

VGG16

Overview: VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in their paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves high accuracy in image recognition tasks by employing a deep architecture consisting of 16 layers (13 convolutional layers and 3 fully connected layers).

Architecture: The network uses very small (3x3) convolution filters throughout the entire architecture, which allows it to capture the finer details of the image. It also uses max pooling layers to reduce the dimensionality of the feature maps.

Strengths: VGG16 is known for its simplicity and its ability to be applied to various image recognition tasks effectively. It has been widely used in the community for benchmarking and developing further enhancements in the field of computer vision.

Read the documentation here:
https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918

MobileNetV2

Overview: MobileNetV2 is a streamlined architecture that is designed for mobile and resource-constrained environments. It builds on the ideas from MobileNetV1, introducing two new features to the architecture: linear bottlenecks and inverted residuals.

Architecture: MobileNetV2 uses depthwise separable convolutions which significantly reduce the number of parameters when compared to a standard convolution with the same output depth. This is achieved by factoring a traditional convolution into a depthwise convolution and a 1x1 convolution called a pointwise convolution.

Strengths: The model is particularly effective in environments where computational resources are limited. It offers a good trade-off between latency, size, and accuracy, making it suitable for real-time applications on mobile devices.

Read the documentation here:
https://vidishmehta204.medium.com/object-detection-using-ssd-mobilenet-v2-7ff3543d738d

Implementation Details

Data Preparation

For this project, focusing on the enhanced recycling process, we refined our approach by utilizing only a subset of the available classes from the Kaggle dataset. This targeted approach involves the following eight classes, chosen for their significant impact on recycling efficiency:

- ➔ Brown-glass
- ➔ Cardboard
- ➔ Clothes
- ➔ Green-glass
- ➔ Metal
- ➔ Paper
- ➔ Plastic
- ➔ White-glass

These classes were selected to represent a broad range of commonly recycled materials, each requiring specific processing techniques, which allows for a more focused and effective training of the neural networks.

Image Resizing: All images were resized to 224x224 pixels, the required input size for both VGG16 and MobileNetV2 models. This uniformity is crucial for maintaining consistency across all inputs to the models.

Normalization: The pixel values of each image were normalized by scaling them to a range between 0 and 1. This step is vital for helping the model process the input more efficiently by standardizing the range of input features.

Data Augmentation: To further enhance the model's ability to generalize and to prevent overfitting, data augmentation techniques were applied using TensorFlow’s ImageDataGenerator. This included random transformations such as rotations, width and height shifts, shear mapping, zooming, and horizontal flipping.

Model Training

VGG16:

Training is conducted with the help of the TensorFlow framework, utilizing the `ImageDataGenerator` to automatically handle image resizing and normalization, thus ensuring the model receives correctly pre-processed images. The model is compiled using the Adam optimizer and sparse categorical cross-entropy loss, which is suitable for multi-class classification tasks. The training process involves only two epochs to demonstrate the setup, though more epochs would typically be used for actual deployment to enhance accuracy. During training, model performance is monitored on a validation set, and the best-performing model is saved using the `ModelCheckpoint` callback, which helps in capturing the most accurate model during the training process. An `EarlyStopping` callback is also employed to halt training if the validation accuracy does not improve for five consecutive epochs, thereby optimizing training time and preventing overfitting. This structured approach ensures the VGG16 model is efficiently adapted to the specific task of garbage classification, leveraging its deep convolutional architecture to discern and categorize various types of household waste effectively.

Analysis:

- ➔ It was very slow when it came to training the model.
- ➔ We ran it for only 2 epochs as it requires a lot of computational power.
- ➔ It gave an accuracy of approximately 80%.

MobileNetV2:

Training the MobileNetV2 model for the task of garbage classification was conducted using the TensorFlow framework, with the `ImageDataGenerator` managing tasks like image resizing and normalization to ensure the input data was properly conditioned for processing. The model, leveraging the efficient architecture of MobileNetV2, was compiled with the Adam optimizer and sparse categorical cross-entropy loss, suitable for multi-class scenarios. The training process was set to run for 15 epochs, but utilized `EarlyStopping` to cease training if no improvement in validation accuracy was observed for five consecutive epochs, thus preventing overfitting and optimizing computational resources. The model's performance was continuously monitored, with `ModelCheckpoint` employed to save the best-performing iterations based on validation accuracy.

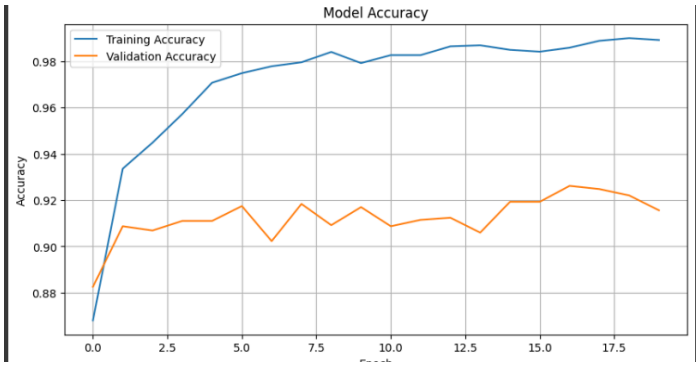
Analysis:

- ➔ It was fast when it came to training the model.
- ➔ We ran it for only 15 epochs but it EarlyStopped at the 13th epoch because it was not improving.

```
273/273 [==============================] - ETA: 0s - loss: 0.0425 - accuracy: 0.9851
Epoch 13: val_accuracy did not improve from 0.92064
273/273 [==============================] - 524s 2s/step - loss: 0.0425 - accuracy: 0.9851 - val_loss: 0.3362 - val_accuracy: 0.9197
Epoch 13: early stopping
```

- ➔ It gave an accuracy of approximately 92.06%.

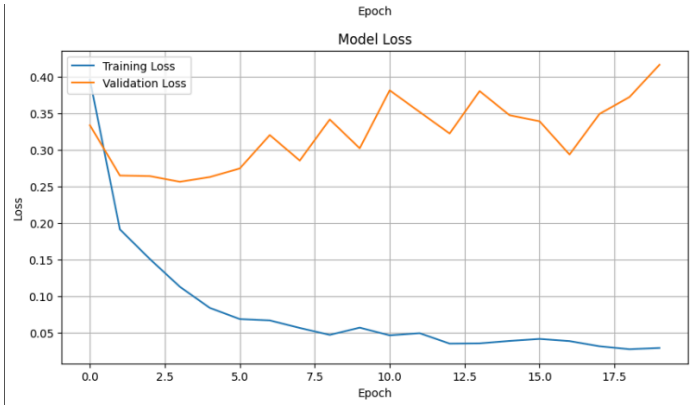
Analysis of Model Accuracy and Loss:



Model Loss:

Training Loss: This decreases sharply initially and then continues to decrease gradually, indicating that the model is effectively learning from the training data over each epoch.

Validation Loss: This starts low but increases over time, suggesting that the model is starting to overfit the training data. Overfitting occurs when a model learns patterns specific to the training data, which do not generalize well to new, unseen data



Model Accuracy:

Training Accuracy: This increases steadily, almost reaching 98%, showing that the model is consistently getting better at correctly predicting the training data.

Validation Accuracy: Although it starts above 90%, it remains relatively flat and even drops slightly over time, further supporting the indication of overfitting seen in the loss graph.

Object Detection Methods:

Real-Time Detection

The real-time object detection was implemented using OpenCV for VGG16 to capture video frames from a webcam. Each frame was processed, resized, and fed into the trained model to predict the object's category. Predictions were displayed on the video feed with a label and a bounding box.

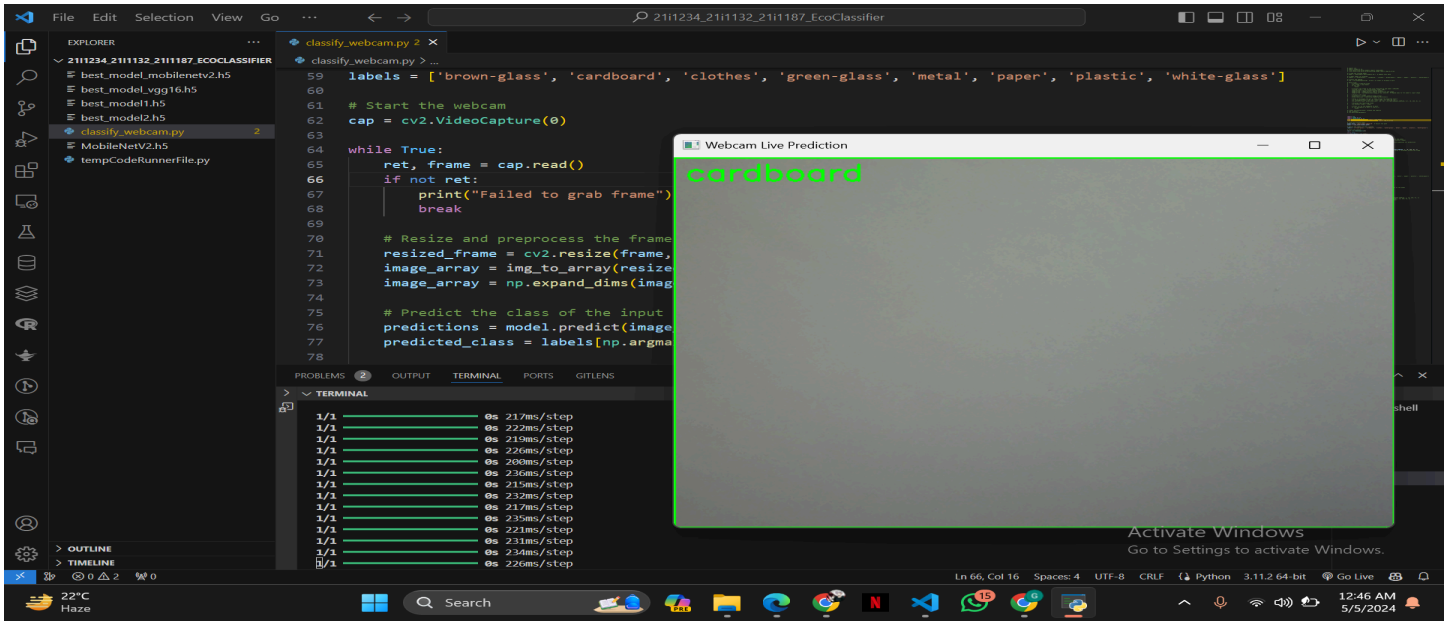
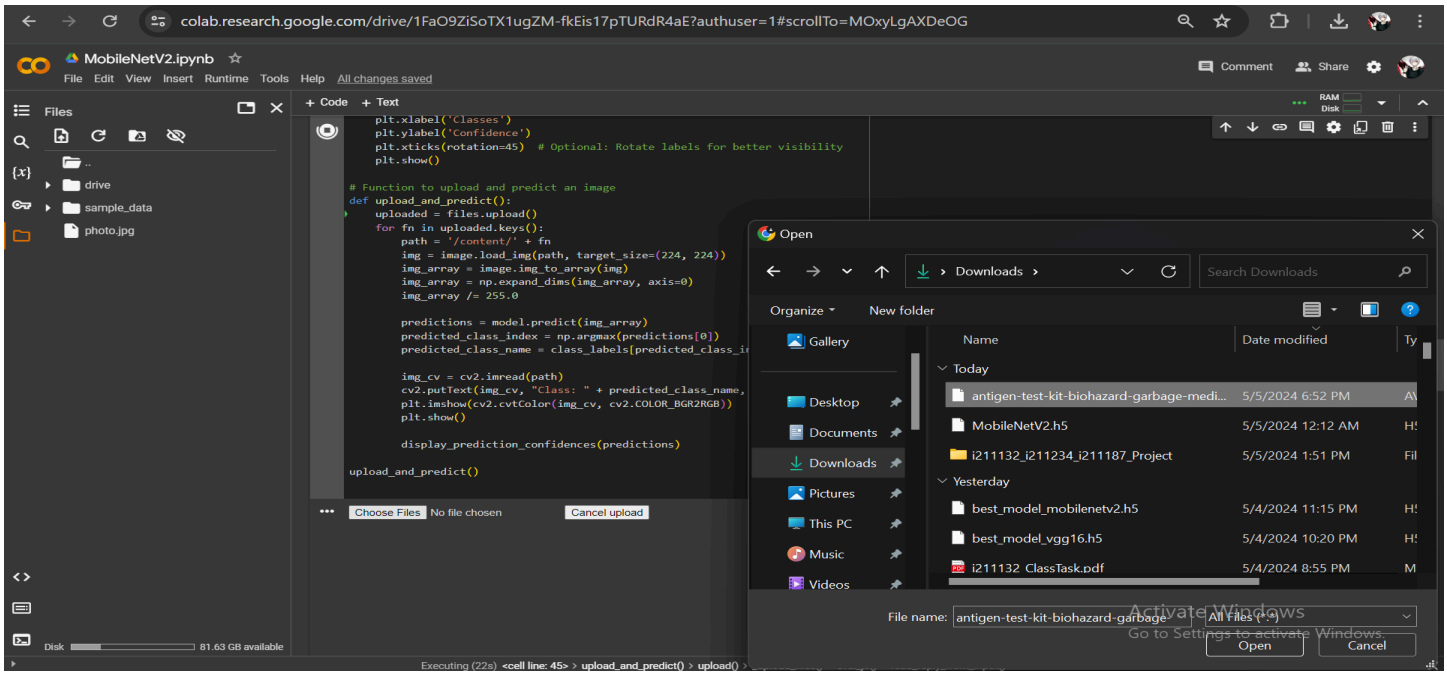


Image Upload Based Detection

The system enables users to upload images directly through the browser or capture them via webcam, which are then processed by the MobileNetV2 model to classify into predefined categories such as 'brown-glass', 'cardboard', and 'clothes', among others. The process involves resizing the images to fit the model's input requirements, classifying them, and displaying both the images and their predicted classifications along with confidence scores for each class.



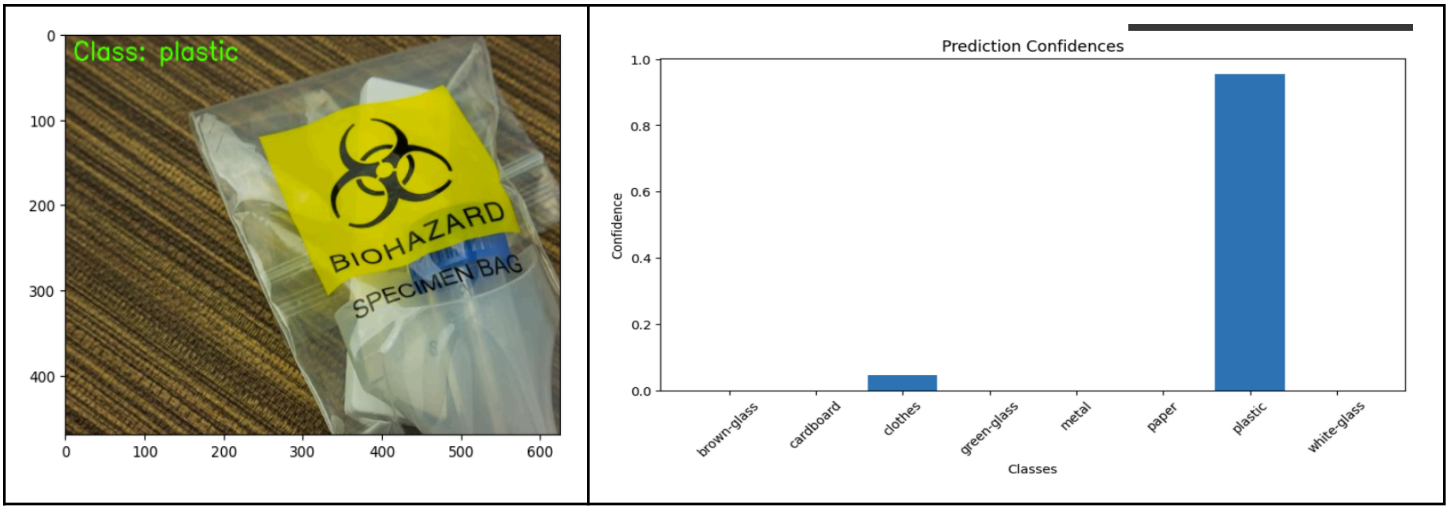
Graph Plotting

The graph, titled "Prediction Confidences," illustrates the confidence levels of a deep learning model in classifying an uploaded image across various predefined categories.

- ➔ The x-axis of the graph represents the different classes that the model can identify, which include 'brown-glass', 'cardboard', 'clothes', 'green-glass', 'metal', 'paper', 'plastic', and 'white-glass'.
- ➔ The y-axis quantifies the confidence level of the model's predictions, ranging from 0 to 1, where 1 signifies absolute certainty in the prediction.

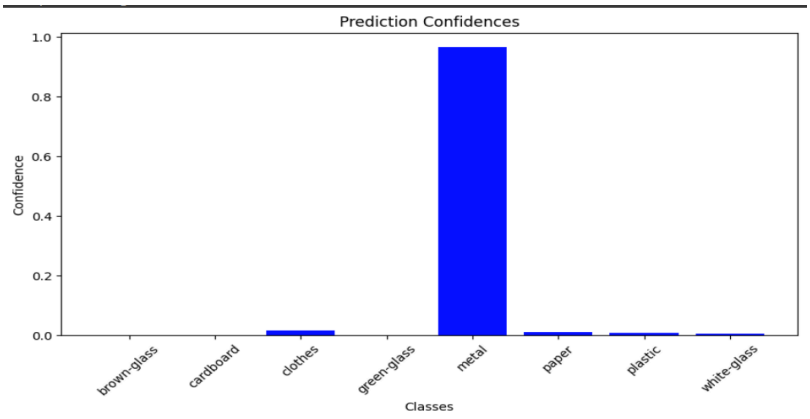
Example - 01

The image uploaded for classification shows a plastic specimen bag with a biohazard symbol. The model has processed the image and assigned the highest confidence score to the "plastic" category, as depicted in the graph where the 'plastic' bar reaches close to 1. This high score implies that the model is almost certain that the item in the image is made of plastic. The other categories show significantly lower confidence scores, indicating that the model has effectively distinguished the material of the item in the image from other possible materials.

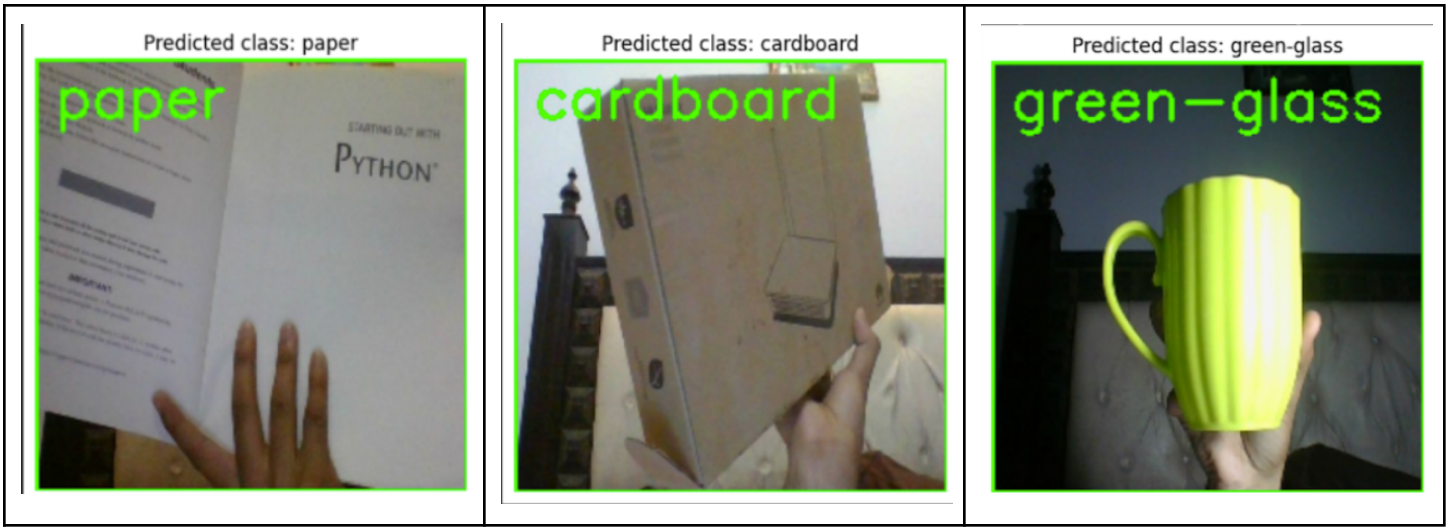


Example - 02

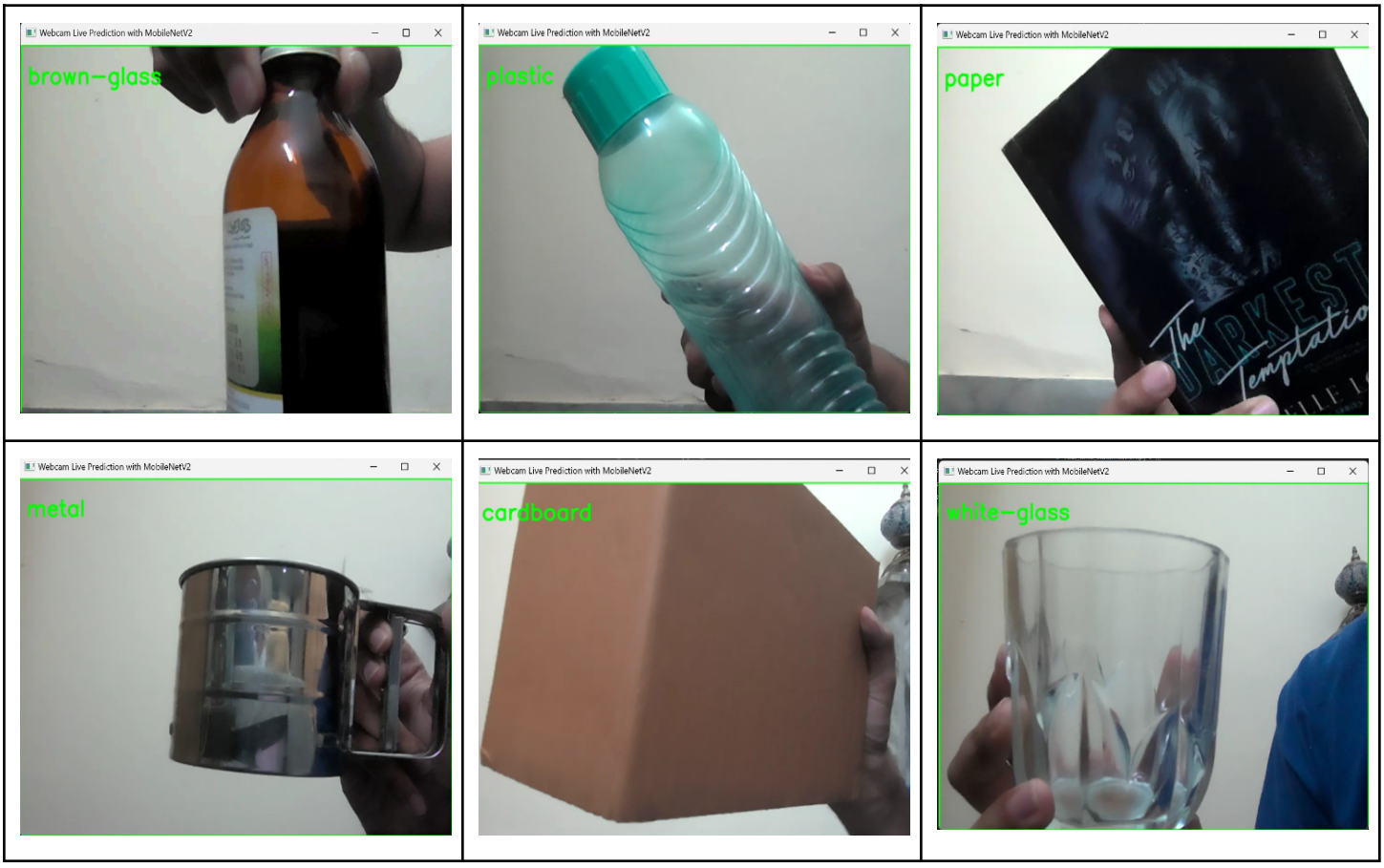
In this particular example, an image of a metal can was uploaded for classification. The confidence graph clearly illustrates that the model is highly confident, almost near certainty, that the object is "metal" as the bar corresponding to the metal class is significantly higher compared to all other classes. The 'metal' bar peaks close to a confidence level of 1.0, indicating a very strong belief by the model that the can is made of metal. The confidence levels for all other material classes are negligible, which suggests the model effectively discriminates between metal and non-metal objects.

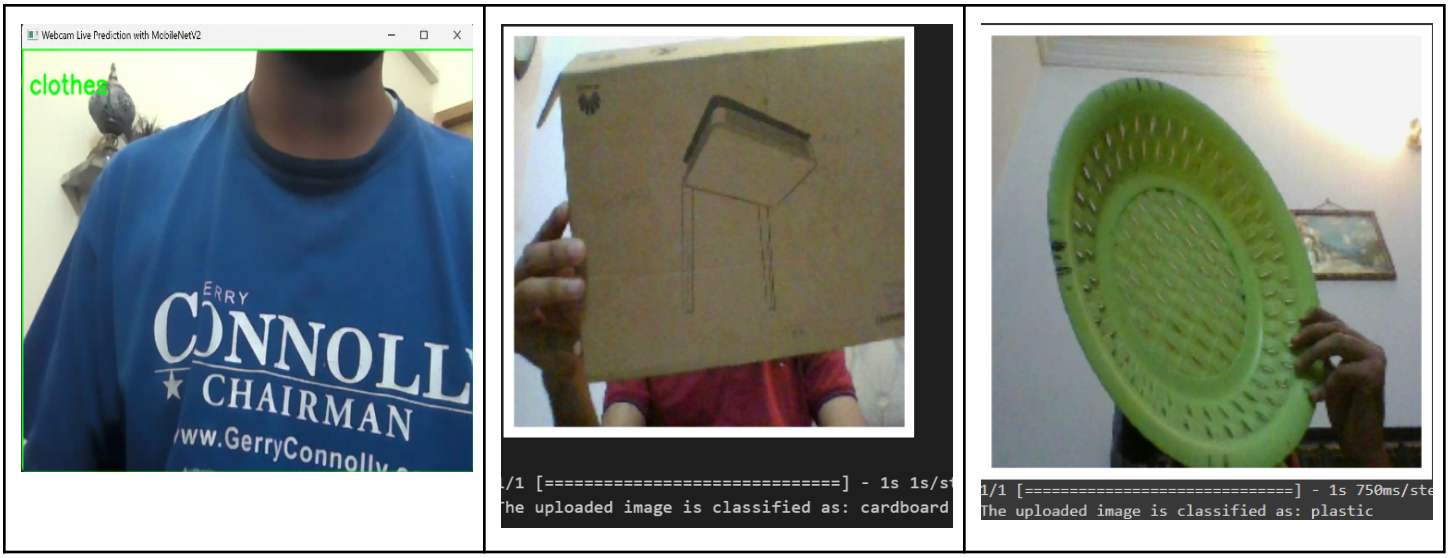


Item Identification: (MobileNet-V2)



Item Identification: (VGG16)





Comparative Analysis

Performance

Accuracy:

In our project, due to computational constraints, VGG16 was limited to just two epochs, achieving an accuracy of approximately 80%. Conversely, MobileNetV2's more efficient architecture allowed for 13 epochs of training, resulting in a higher accuracy of about 92%. This demonstrates MobileNetV2's suitability for scenarios with limited computational resources, where it can still perform effectively and efficiently.

Speed and Efficiency: MobileNetV2 is significantly faster and more efficient than VGG16 due to its lightweight architecture. This makes it more suitable for real-time object detection in a live video stream using a webcam.

Use Case Suitability

VGG16: More suitable for applications where high accuracy is paramount and computational resources are sufficient.

MobileNetV2: Ideal for mobile applications and real-time video processing where computational resources and power consumption are limited.

Challenges:

Computational Resource Constraints
One of the primary challenges encountered in this project was the significant demand on computational resources posed by the VGG16 model. VGG16, with its deep convolutional architecture, requires substantial computational power and memory for processing, which can strain the capabilities of ordinary laptops and desktops. The intensive nature of the computations, especially during the training phase involving large datasets and image transformations, leads to prolonged training times and decreased efficiency.

Dependency on Stable Internet Connection
Another significant hurdle was the dependency on a stable internet connection, particularly critical when using cloud-based platforms like Google Colab for training the models. The training process for deep learning models, especially ones as large as VGG16, can take several hours or even days. Any interruption in the internet connection during this period could lead to a halt in training, resulting in loss of progress and wasted computational resources. This requirement for continuous internet connectivity adds an extra layer of complexity in managing the training sessions, especially in environments where internet services are unreliable.

Duration of Model Training
The lengthy duration required to train complex models like VGG16 also posed a challenge. Extensive training periods are not just time-consuming; they also tie up computational resources for the duration, limiting the ability to perform other tasks and increasing the cost of electricity and possibly cloud computing time. This extended duration is necessary to achieve the high levels of accuracy expected of such models but can be impractical for rapid development cycles or situations where quick deployment of updated models is required.

Conclusion

The project successfully demonstrates the application of VGG16 and MobileNetV2 in real-time object detection. Each model offers unique advantages, with VGG16 providing high accuracy and MobileNetV2 offering greater efficiency. Depending on the specific requirements of an application, either model can be chosen to balance between accuracy and performance.