Computer Vision

ASSIGNMENT:02

Due Date: 27th-Oct- 2024

Instructions:

- This assignment is a group project, and each group must consist of **2 members**.
- Submit your assignment by **27th-Oct- 2024**. Late submissions will incur a **penalty**. No submissions will be accepted after **Due Time** without prior approval.
- Your work must be your own. You may discuss ideas with peers but do not share code. Plagiarism will result in serious consequences, including a failing grade.
- Ensure your code is clean, readable, and well-organized. Use meaningful variable names and comments. Include a **README** explaining how to run your code.
- Test your code thoroughly. Handle edge cases and report any known issues in the README.
- Submit through **GCR**. Include your code, README, and other required deliverables in a **.zip** or **.tar.gz** file.
- If you have questions, reach out during office hours or help sessions, but don't wait until the last minute.
- Grading will be based on correctness, code quality, efficiency, and adherence to the instructions.

Automating Hidden Object Games with Object Detection

Remember the thrill of playing **hidden object games**, where you have to find cleverly concealed items in a cluttered scene? These games challenge our perception, making us search carefully through piles of objects, messy rooms, or crowded environments. But what if we could automate this process? Imagine having an intelligent system that can instantly spot and identify hidden objects, solving the game in seconds!

In this assignment, your task is to **build an automated solution** to this problem using the **object detection algorithms** studied in the course. You'll apply techniques like YOLO or other detection models to find objects in complex, cluttered scenes, similar to those in hidden object games.

Requirements

1. Technical Requirements:

- **Dataset Selection and Fine-Tuning**:

- Students must fine-tune **2 object detection models** using **2 different datasets** from the following: **iMaterialist, Object 365, or NUScenes**.

  iMaterialist: https://www.kaggle.com/c/imaterialist-fashion-2020-fgvc7

  Object365: https://www.objects365.org/overview.html

  NUScenes: https://www.nuscenes.org/

- **Dataset Size**: Each dataset is quite large, so students are required to **reduce the size** by creating a balanced subset.
- **Subset Criteria**:
  - The subset should include at least **10 classes** or more, with a total dataset size of **50,000 images**.
  - **Proper sampling** should be done to ensure that the classes are balanced.
- Students must explain how they selected the subset in their report (see Reporting Requirements below).

- **Object Detection Algorithms**:
  - Students should experiment with **two different object detection algorithms** (e.g., YOLO, Faster R-CNN, RetinaNet, etc.).
  - Models should be fine-tuned specifically for the chosen datasets.

2. Game-Related Requirements (Frontend Features):

- **Frontend Interface**:
  - Students must build a simple frontend interface, which may utilize existing templates from the internet.
  - **Frontend Features**:
    - **Dropdown Menus**: For selecting the dataset and object detection model. These should be labeled clearly, e.g., datasets could be presented as "scenes" for the hidden object game, while the model names can be chosen by the student.
    - **Image Upload**: Allow the user to upload an image for object detection.
    - **Image Display**: The uploaded image should be displayed in a container window.
    - **"Find" Button**: This button triggers the object detection inference. Detected objects should be shown using bounding boxes.
    - **Score Display**: A text box should display the score, comparing the total number of objects found with the total objects in the image according to the dataset's annotations.
    - **Inference Time Display**: Another text box should display the inference time for the model to detect objects in the image.

3. Reporting Requirements:

- **Dataset Selection**:
  - List the two datasets selected in bullet points.
  - Include a brief explanation of how the subsets were created.
- **Data Subset Criteria and Distribution**:
  - Explain the criteria used to select the subset (e.g., class balancing, number of images per class, etc.).
  - Include a **graph** of the data distribution over the selected classes to show whether the data is balanced.
- **Model Selection and Reasoning**:
  - Write a short paragraph explaining why you chose the two specific object detection models for the assignment.
- **Comparative Analysis**:
  - Compare the performance of the two models on both datasets.
  - Use **tables and graphs** to present the results (e.g., precision, recall, accuracy, inference time).
  - Provide a brief explanation for the results displayed in the tables and graphs.
- **Insights and Observations**:
  - Share any **observations** about the model's performance. If a model underperformed, explain potential reasons (e.g., complexity, overfitting, dataset imbalance).
  - Include any **challenges** encountered during the implementation and fine-tuning process, along with solutions
  -
- **Screenshots of Front-End:**
  - Add screenshots of the front end showing the working of its features.