Report By Raghav Mehta (12319917) for submission of Online assignment on Image & Object detection round of SafetyWhat .

1. Introduction

The purpose of this report is to provide a detailed analysis of the inference speed, system architecture, and optimization strategies used in the object detection system. This analysis is critical for evaluating the system's real-time performance, scalability, and efficiency.

---

2. Inference Speed Results

Inference speed is an important factor in real-time applications such as object detection, where the system must process multiple frames per second with minimal latency.

Key Metrics:

- **Average Inference Time per Frame:** 0.1191 seconds
- **Inference Speed (FPS):** 8.40 FPS

These results indicate that the system is capable of processing approximately 8.4 frames per second, with each frame taking around 0.1191 seconds to analyze. The relatively high FPS ensures the system can handle real-time data, such as video streams, with low latency.

Object Detection Performance:

The object detection model was evaluated on its ability to count and identify specific object categories within each frame. The results are as follows:

- **Person:** 176 instances
- **Car:** 868 instances
- **Motorcycle:** 214 instances
- **Truck:** 173 instances

The system demonstrated its capability to detect a diverse range of objects across multiple categories, contributing to its effectiveness in applications such as surveillance or autonomous vehicles.

---

3. System Architecture

The architecture of the object detection system was designed to ensure high performance and efficiency, from data processing to model inference. Below are the main components of the system:

3.1 Data Preprocessing Layer:

This layer is responsible for handling raw data inputs, including preprocessing steps like resizing images, normalizing pixel values, and augmenting data as required. These transformations help prepare the data for input into the model.

3.2 Model Inference Engine:

The core of the system, where the trained object detection model performs inference. The model was selected for its ability to balance accuracy and speed. Inference is done in real-time for each frame, providing immediate predictions.

3.3 API Layer:

This layer interacts with external systems or users, accepting input data (images or video frames), invoking the inference engine, and returning the results. It was built using [insert framework, e.g., Flask, FastAPI], ensuring smooth integration with other applications.

3.4 Data Storage Layer:

Stores model weights, configuration files, and logs of inference results. A database or cloud storage solution, such as [insert database or storage system], ensures that all model metadata and results are securely stored for future reference.

---

4. Optimization Strategies

To improve inference speed and ensure the system performs optimally, several optimization strategies were employed:

4.1 Model Pruning:

Unimportant connections and neurons within the model were pruned to reduce its size, allowing for faster processing while maintaining accuracy.

4.2 Quantization:

Quantizing the model weights from 32-bit floating point to lower precision (e.g., 8-bit integers) reduces both the memory footprint and computation time, accelerating inference speed.

4.3 Hardware Acceleration:

The system leveraged [insert hardware details, e.g., GPU/TPU] to accelerate model inference. Hardware acceleration provides a significant speedup, especially in complex models requiring high computational power.

4.4 Batching:

Instead of processing individual frames separately, multiple frames were batched together for inference. This batch processing reduces overhead and improves throughput, particularly in high-volume environments.

4.5 TensorRT/ONNX Optimizations:

The model was exported to the ONNX format and optimized using TensorRT. This optimization framework enhances the speed of deep learning models by utilizing low-level hardware optimizations.

4.6 Caching:

To avoid redundant calculations, repeated queries were cached. This caching mechanism is particularly useful for scenarios where similar data is processed repeatedly, enhancing overall efficiency.

---

5. Conclusion

The optimizations implemented have significantly improved the system's inference speed, allowing it to process video frames at 8.4 FPS. The object detection system is now capable of detecting and counting various objects in real-time, with the system's architecture and optimization strategies ensuring high efficiency and low latency.

---

6. Future Work

Future work will focus on further improving the inference speed by exploring additional optimization techniques such as distributed processing and exploring other model architectures. Moreover, enhancing the system's accuracy, particularly in more complex object detection tasks, will remain a priority.

---

7. Appendix (Optional)

- **Hardware Specifications:**
  - Processor: [insert processor details]
  - RAM: [insert RAM details]
  - GPU: [insert GPU details]
  - Storage: [insert storage details]
- **Software Details:**
  - Operating System: [insert OS]
  - Framework: [insert framework, e.g., TensorFlow, PyTorch]
  - Model Used: [insert model details]