OCR Model Conversion: GPU to CPU

Company: Tensorie Software Pvt Ltd

Objective:
Convert an existing GPU-based OCR (Optical Character Recognition) model to a CPU-based version while maintaining or improving accuracy and FPS (Frames Per Second).

Technology Stack:

1. **Frameworks/Libraries:**
   - **TensorFlow:**For developing, training, and optimizing neural networks. TensorFlow Lite is recommended for model conversion to run efficiently on CPUs.
     - TensorFlow
   - **PyTorch:**A deep learning framework that allows for easy model optimization and deployment on CPUs.
     - PyTorch
   - **OpenVINO Toolkit:**For optimizing and deploying models on Intel CPUs and accelerators.
     - OpenVINO
   - **ONNX Runtime:**A cross-platform engine for optimized CPU inference using ONNX models.
     - ONNX Runtime
2. **Tools:**
   - **FFmpeg:**A multimedia framework for handling and processing video files.
     - FFmpeg
   - **Jupyter Notebook:**For interactive model development and testing.
     - Jupyter Notebook
   - **Python:**The primary programming language for working with deep learning models.
     - Python
3. **Optimization Techniques:**
   - **Quantization:**Reduce precision to optimize performance on the CPU.
     - TensorFlow: Quantization with TensorFlow Lite
     - PyTorch: Quantization in PyTorch
   - **Pruning:**Remove unnecessary weights to make the model lighter.
     - TensorFlow: Model Pruning
     - PyTorch: Model Pruning in PyTorch
4. **Performance Evaluation:**
   - **Profiling Tools:**Use TensorFlow Profiler or PyTorch's built-in tools for evaluating performance.
     - TensorFlow: TensorFlow Profiler
     - PyTorch: Profiling in PyTorch

Steps to Follow:

1. Model Conversion

- **Select the Existing OCR Model:** Identify the GPU-based OCR model to be converted.
- **Optimize for CPU:** Use quantization, pruning, and batch size adjustment to optimize the model for CPU processing. Convert the model using TensorFlow Lite, OpenVINO, or ONNX Runtime.

2. Performance Evaluation

- **Accuracy Check:** Ensure the OCR outputs' accuracy is maintained or improved.
- **Speed (FPS) Measurement:** Record FPS for both GPU and CPU models.
- **Resource Utilization:** Monitor CPU and memory usage during the evaluation.

3. Input/Output Processing

- **Video Processing:** Use FFmpeg to process the provided video file with both GPU and CPU models.
- **Create Demonstration Content:**
  - **1-Minute Visual Comparison:** Simulate a side-by-side comparison of GPU vs. CPU model performance.
  - **2-Minute Theoretical Explanation:** Explain the conversion techniques, strategies to maintain accuracy and FPS, and challenges encountered.

Deliverables

1. Converted OCR Model:

- A fully optimized OCR model for CPU with minimal or no accuracy drop.

2. Processed Video File:

- A video showcasing the performance of both GPU and CPU models.

3. Comparative Analysis Report:

- A detailed analysis comparing accuracy, FPS, and resource utilization.

4. 3-Minute Demonstration Video:

- **1-Minute:** Conceptual showcase of GPU vs. CPU model performance.
- **2-Minutes:** Explanation of the conversion process, optimization strategies, and challenges.

Evaluation Criteria

- **Model Performance:** Successful conversion of the model to CPU with minimal loss in accuracy and maintained or improved FPS.
- **Clarity and Quality:** Clear and well-structured comparison and explanation in the demonstration video.

Resources:

1. **TensorFlow Model Optimization:**
   - TensorFlow Lite Model Optimization Guide
   - TensorFlow for Mobile and Edge Devices
2. **PyTorch Model Optimization:**
   - Quantization in PyTorch
   - TorchScript for Optimizing Models
3. **OpenVINO for CPU Optimization:**
   - OpenVINO Documentation
   - OpenVINO Model Optimizer
4. **Video Processing with FFmpeg:**
   - FFmpeg Documentation
   - FFmpeg Basic Commands