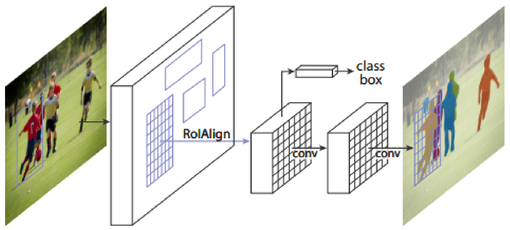
Optimizing MASK-RCNN-FPN with Quantization

Haoyu Chen [1] Hassan Iqbal [2] Muhammad Shahzad [3]

[1]Xidian University [2]Supervisor, NC State[3]Mentor, NC State

Abstract

The issue lies in the heavy size and slow processing speed of the **MASK-RCNN-FPN model** trained on detectron2, hindering efficient object detection.



This study proposes a two-step solution to optimize Detectron2's object detection model:

- **converting the model into a TorchScript representation**
- **applying dynamic quantization techniques to compress parameters**

Motivation

- **optimizing the speed** optimization plays a crucial role in accurately predicting and optimizing the speed of cloud gaming platforms.
- **enhancing visual experience** Through quantitative optimization of video stream delivery, consistent and high-quality gaming streams can be ensured.

Tasks and Methods

• Implement quantization to optimize model

Research and select quantization methods suitable for optimizing the MASK-RCNN-FPN model.Implement the chosen quantization methods to reduce model size and improve prediction speed.

• Evaluate optimized model's performance

Compare the performance of the optimized model to the original model by measuring prediction time, average precision, detection accuracy, and speed.

Experiments

1. Data Preprocessing and Untuned Model Training:

- **Data Preprocessing:** Using scripts to convert JSON to COCO format, preparing the data for MASK-RCNN-FPN training.
- **Training and Analysis of model:** Initializing, training, and evaluating the performance of the model for object detection.

2. TorchScript Conversion & Dynamic Quantization:

- **Detectron2's Limitation:** Identifying the absence of quantization capabilities.
- **TorchScript Conversion:** To integrate quantization techniques, effectively improve the model efficiency, transforming the PyTorch model into TorchScript model.
- **Dynamic Quantization:** Compressing the model's parameters into 8-bit integers, guided by input data analysis.

3. Efficient Prediction and Evaluation Framework with TorchScript:

- **Custom TorchScriptPredictor Class Creation:** This class is created to handle TorchScript models, as DefaultPredictor from Detectron2 is unable to process them. It inherits from DefaultPredictor, preprocesses images and improves prediction efficiency on the GPU.
- **Iterative Inference, Post-processing, and Time Tracking:** It executed iterations on testing data, including producing quantized outputs, refining predictions, adjusting masks, and tracking inference time. This allows for insights into the prediction acceleration of the quantized model.

Results

No Quantization	With Quantization
0.164	0.132

Table 1. Average Prediction Time for every Image

No Quantization	With Quantization
70.1	62.4

Table 2. Average Precision (AP) for boxes

References

[1] Hassan Iqbal, Ayesha Khalid, and Muhammad Shahzad. Dissecting cloud gam-ing performance with decaf. Proc. ACM Meas. Anal. Comput. Syst.5(3), dec 2021.

[2] He, Kaiming, et al. "Mask r-cnn." Proceedings of the IEEE international conference on computer vision. 2017.

My project: https://github.com/Max-chy/GEARS-Quantization

haoyu_24@136.com