Abstract

oneDNN Graph library extends oneDNN with a flexible graph API to maximize the optimization opportunities for generating efficient code on AI hardware.

oneDNN Graph automatically identifies the graph partitions to be accelerated via fusion. Its fusion patterns entail fusing compute-intensive operations such as convolution, matmul and their neighbor operations for both inference and training use cases.

- Since PyTorch 1.12, oneDNN Graph fuser has been supported as an experimental feature to speed up inference with Float32 datatype or x86-64 CPUs.
- Support for inference with oneDNN Graph using BFloat16 datatype exists in the PyTorch master branch, and hence also in nightly PyTorch releases.
- Intel Extension for PyTorch is an open-source library that builds on top of PyTorch, and can be thought of as a "staging-ground" for optimizations in PyTorch from Intel. It leverages oneDNN Graph for inference with int8 datatype.

Overview of oneDNN Graph

Performance comparison using TorchBench

Multi-threaded performance with 26 threads on all the 26 physical cores of one socket of Intel Xeon Platinum 8371HC (3rd gen Xeon SP)

- Speedup of FP32 Inference with PyTorch & oneDNN Graph over NNC OFI FP32
- Speedup of BF16 Inference with PyTorch & oneDNN Graph over NNC OFI FP32
- Speedup of INT8 Inference with Intel Extension for PyTorch & oneDNN Graph over NNC OFI FP32

 Currently supported fusion patterns in PyTorch with oneDNN Graph

- Convolution Post-ops
  - Pattern: Convolution + BiasAdd + BatchNormInference + [Unary | Binary]
  - Description: this pattern is widely used in Convolution Neural Networks, i.e. ResNet, ResNext, SSD, etc.

- MatMul Post-ops
  - Pattern: MatMul + BiasAdd + [Unary | Binary]
  - Description: this pattern is widely used in language models and recommendation models, i.e. BERT, DLRM, etc.

- Reduction Post-ops
  - Pattern: Reduction + [Unary | Binary]
  - Description: this pattern is widely used for data processing, i.e. loss reduction.

- Unary Post-ops
  - Pattern: Unary + Binary
  - Description: this pattern is widely used in Convolution Neural Networks.

- Binary Post-ops
  - Pattern: Binary + [Unary | Binary]
  - Description: this pattern is widely used in Generative Adversarial Networks, i.e. ParallelWaveGAN.

- Pooling Post-ops
  - Pattern: [AvgPool | MaxPool] + Binary
  - Description: this pattern is widely used in Convolution Neural Networks.

- Batch Normalization Post-ops
  - Pattern: BatchNormInference + ReLU
  - Description: this pattern is widely used in Convolution Neural Networks, i.e. DenseNet.

### Acknowledgements

oneDNN Graph integration with PyTorch was done with the advice & guidance of Elias Ellison (Meta), Jian Hui Li (Intel Corporation), Chunyuan Wu (Intel Corporation), Jiong Gong (Intel Corporation).