Huggingface QDQBERT Quantization Example

The QDQBERT model adds fake quantization (pair of QuantizeLinear/DequantizeLinear ops) to: * linear layer inputs and weights * matmul inputs * residual add inputs

In this example, we use QDQBERT model to do quantization on SQuAD task, including Quantization Aware Training (QAT), Post Training Quantization (PTQ) and inferencing using TensorRT.

Required: - pytorch-quantization toolkit - TensorRT >= 8.2 - PyTorch >= 1.10.0

Setup the environment with Dockerfile

Under the directory of `transformers/`, build the docker image:

```
docker build . -f examples/research_projects/quantization-qdqbert/Dockerfile -t bert_quantiz
```

Run the docker:

```
docker run --gpus all --privileged --rm -it --shm-size=1g --ulimit memlock=-1 --ulimit stack
```

In the container:

```
cd transformers/examples/research_projects/quantization-qdqbert/
```

Quantization Aware Training (QAT)

Calibrate the pretrained model and finetune with quantization awared:

```
python3 run_quant_qa.py \
  --model_name_or_path bert-base-uncased \
  --dataset_name squad \
  --max_seq_length 128 \
  --doc_stride 32 \
  --output_dir calib/bert-base-uncased \
  --do_calib \
  --calibrator percentile \
  --percentile 99.99
```

```
python3 run_quant_qa.py \
  --model_name_or_path calib/bert-base-uncased \
  --dataset_name squad \
  --do_train \
  --do_eval \
  --per_device_train_batch_size 12 \
  --learning_rate 4e-5 \
  --num_train_epochs 2 \
  --max_seq_length 128 \
```

```
  --doc_stride 32 \
  --output_dir finetuned_int8/bert-base-uncased \
  --tokenizer_name bert-base-uncased \
  --save_steps 0
```

Export QAT model to ONNX

To export the QAT model finetuned above:

```
python3 run_quant_qa.py \
  --model_name_or_path finetuned_int8/bert-base-uncased \
  --output_dir ./ \
  --save_onnx \
  --per_device_eval_batch_size 1 \
  --max_seq_length 128 \
  --doc_stride 32 \
  --dataset_name squad \
  --tokenizer_name bert-base-uncased
```

Use `--recalibrate-weights` to calibrate the weight ranges according to the quantizer axis. Use `--quant-per-tensor` for per tensor quantization (default is per channel). Recalibrating will affect the accuracy of the model, but the change should be minimal (< 0.5 F1).

Benchmark the INT8 QAT ONNX model inference with TensorRT using dummy input

```
trtexec --onnx=model.onnx --explicitBatch --workspace=16384 --int8 --shapes=input_ids:64x128
```

Evaluate the INT8 QAT ONNX model inference with TensorRT

```
python3 evaluate-hf-trt-qa.py \
  --onnx_model_path=./model.onnx \
  --output_dir ./ \
  --per_device_eval_batch_size 64 \
  --max_seq_length 128 \
  --doc_stride 32 \
  --dataset_name squad \
  --tokenizer_name bert-base-uncased \
  --int8 \
  --seed 42
```

Fine-tuning of FP32 model for comparison

Finetune a fp32 precision model with transformers/examples/pytorch/question-answering/:

```
python3 ../../pytorch/question-answering/run_qa.py \
```

```
  --model_name_or_path bert-base-uncased \
  --dataset_name squad \
  --per_device_train_batch_size 12 \
  --learning_rate 3e-5 \
  --num_train_epochs 2 \
  --max_seq_length 128 \
  --doc_stride 32 \
  --output_dir ./finetuned_fp32/bert-base-uncased \
  --save_steps 0 \
  --do_train \
  --do_eval
```

Post Training Quantization (PTQ)

PTQ by calibrating and evaluating the finetuned FP32 model above:

```
python3 run_quant_qa.py \
  --model_name_or_path ./finetuned_fp32/bert-base-uncased \
  --dataset_name squad \
  --calibrator percentile \
  --percentile 99.99 \
  --max_seq_length 128 \
  --doc_stride 32 \
  --output_dir ./calib/bert-base-uncased \
  --save_steps 0 \
  --do_calib \
  --do_eval
```

Export the INT8 PTQ model to ONNX

```
python3 run_quant_qa.py \
  --model_name_or_path ./calib/bert-base-uncased \
  --output_dir ./ \
  --save_onnx \
  --per_device_eval_batch_size 1 \
  --max_seq_length 128 \
  --doc_stride 32 \
  --dataset_name squad \
  --tokenizer_name bert-base-uncased
```

Evaluate the INT8 PTQ ONNX model inference with TensorRT

```
python3 evaluate-hf-trt-qa.py \
  --onnx_model_path=./model.onnx \
  --output_dir ./ \
  --per_device_eval_batch_size 64 \
  --max_seq_length 128 \
```

```
--doc_stride 32 \
--dataset_name squad \
--tokenizer_name bert-base-uncased \
--int8 \
--seed 42
```

Quantization options

Some useful options to support different implementations and optimizations. These should be specified for both calibration and finetuning.

argument	description
`--quant-per-tensor`	quantize weights with one quantization range per tensor
`--fuse-qkv`	use a single range (the max) for quantizing QKV weights and output activations
`--clip-gelu N`	clip the output of GELU to a maximum of N when quantizing (e.g. 10)
`--disable-dropout`	disable dropout for consistent activation ranges