AIMET

Quantization Aware Training Example Code

Quantization simulation and finetuning using the AIMET library.

The general procedure for quantization is to use AIMET's QuantizationSimModel to compute new encodings, then finetune the model.

The weights of the pretrained model (in our case, ResNet), are originally 32-bit floating point numbers, which are then converted to 8-bit numbers through a rounding procedure.

Then, this quantized model can be finetuned.

This script utilizes AIMET to perform Quantization Aware Training on a ResNet18 pretrained model with the ImageNet data set.

Scenario parameters:

- AIMET quantization aware training using simulation model

```
from aimet_torch.quantsim import QuantizationSimModel
```

- Quant Scheme: 'tf'
- rounding_mode: 'nearest'
- default_output_bw: 8, default_param_bw: 8
- Encoding computation using 5 batches of data
- Input shape: [1, 3, 224, 224]
- Learning rate: 0.001
- Decay Steps: 5

1. Instantiate Data Pipeline for evaluation
2. Load the pretrained ResNet18 Pytorch model
3. Calculate Model accuracy
    a. 3.1. Calculate floating point accuracy
    b. 3.2. Calculate Quant Simulator accuracy
4. Apply AIMET CLE and BC
    a. 4.1. Apply AIMET CLE and calculates QuantSim accuracy
    b. 4.2. Apply AIMET BC and calculates QuantSim accuracy
5. Fine-tune Model

Setting Up Our Config Dictionary

The config dictionary specifies a number of things

config: This mapping expects following parameters:

- **dataset_dir:** Path to a directory containing ImageNet dataset. This folder should contain subfolders 'train' for training dataset and 'val' for validation dataset.
- **use_cuda:** A boolean var to indicate to run the quantization on GPU.
- **logdir:** Path to a directory for logging.

To get a better understanding of when each of the parameters in the config dictionary is used, read the code in those cells.
Note: You will have to replace the dataset_dir path with the path to your own imagenet/tinyimagenet dataset

1. Instantiate Data Pipeline

The ImageNetDataPipeline class takes care of evaluating a model using a dataset directory.

The data pipeline class is simply a template for the user to follow. The methods for this class can be replaced by the user to fit their needs.

```
ImageNetDataPipeline

Provides APIs for model quantization using evaluation and finetuning.


data_loader

return: ImageNetDataloader

evaluate

Evaluate the specified model using the specified number of samples from the validation
set.        :param model: The model to be evaluated.        :param iterations: The number
of batches of the dataset.        :param use_cuda: If True then use a GPU for inference.
:return: The accuracy for the sample with the maximum accuracy.

finetune

Finetunes the model.  The implemtation provided here is just an example,        provide
your own implementation if needed.        :param model: The model to finetune.
```

2.Load the Model, Initialize DataPipeline

initialize the pipeline and the model. Before quantizing the model, we calculate the original floating point (FP32) accuracy of the model on the dataset provided.

3. Quantization Simulator

Quantization Parameters

- **quant_scheme**: The scheme used to quantize the model. We can choose from s - post_training_tf or post_training_tf_enhanced.
- **rounding_mode**: The rounding mode used for quantization. There are two possible choices here - 'nearest' or 'stochastic'
- **default_output_bw**: The bitwidth of the activation tensors. The value of this should be a power of 2, less than 32.
- **default_param_bw**: The bidwidth of the parameter tensors. The value of this should be a power of 2, less than 32.
- **num_batches**: The number of batches used to evaluate the model while calculating the quantization encodings.Number of batches to use for computing encodings. Only 5 batches are used here to speed up the process. In addition, the number of images in these 5 batches should be sufficient for compute encodings

3.1 Calculate floating point accuracy

3.2. Calculate Quant Simulator accuracy

`fold_all_batch_norms and get the BN_folded_model`

`Quantize the BN_folded_model using QuantizationSimModel method`

`compute_encodings`

`Evaluate accuarcy`

4. Apply AIMET CLE and BC

4. 1 Cross Layer Equalization

`Call equalize_model`

`This API will equalize the model in-place`

Then, the model is quantized, and the accuracy is noted. This is done before the bias correction step in order to measure the individual impacts of each technique.

Calculate `CLE applied Model Top-1 accuracy on Quant Simulator`

4. 2 Bias Correction

Perform Bias correction and calculate the accuracy on the quantsim model.

- **num_quant_samples**: The number of samples used during quantization
- **num_bias_correction_samples**: The number of samples used during bias correction

5. Fine-tune Model

After the model is quantized, the model is finetuned, then evaluated and saved.

```
# AIMET Imports for Quantizationfrom aimet_common.defs import QuantScheme

Quantization Aware Training Library

from aimet_torch.quantsim import QuantizationSimModel, QuantParamsfrom
aimet_torch.bias_correction import correct_biasfrom aimet_torch.cross_layer_equalization
import equalize_modelfrom aimet_torch.batch_norm_fold import fold_all_batch_norms
```

References

https://github.com/quic/aimet/blob/develop/Examples/torch/quantization/quantization_aware_training.ipynb

https://github.com/quic/aimet/blob/a6b582f83f557c58ae38289a9977a009dd7cc2ec/Examples/torch/quantization/quantization_aware_training.py