18-643 Recitation 4

Vitis AI

Chengyue Wang

Fall 2021
Introduction to Vitis AI

- Accelerating AI inference on hardware platform (FPGA / ACAP)
- Capable of a large range of deep learning tasks
- Easy of use by supporting mainstream AI frameworks (Caffe/TensorFlow/PyTorch)
Overview

User Applications

Frameworks

Well Trained Models

- Xilinx Model Zoo
- Public Model Zoo
- Customized Models

Xilinx IR

- AI Parser
- AI Quantizer

Xilinx Compiler

- AI Compiler

Xilinx Embedded Software

- AI Library
- AI Runtime

Xilinx Runtime

Deep Learning Processing Unit (DPU)

- CNN-Zynq
- CNN-Alveo
- LSTM-Alveo
- CNN-AIE
- LSTM-AIE
- ...

Picture from Xilinx
Quantizer:

• Quantize float parameters to fixed point precision for better performance
• Calibration to control performance loss

> Uniform symmetric quantization
  > 8bit for both weights and activation

> Support both calibration and finetune
  > Calibration – A small set of training data
  > Finetune – Original training data, further increase accuracy

> Support framework
  > Caffe, Tensorflow
  > Pytorch (Q2, 2020)

> Have both GPU and CPU version
  > GPU version is 10x faster than CPU version
Compiler:
- Map models to highly optimized DPU instruction sequences
- Map DPU unsupported OPs to CPU
- Optimization / OP fusion

VART:
- runtime Python/C APIs to start/stop running model on the board
  ex: execute_async() create_runner()

DPU: Deep Learning Processing Unit
- Configurable computation engine optimized for neural network
  CNN/LSTM/MLP/YOLO/SSD/VGG...
- (Domain-specific Architecture)
On Ubuntu 18.04 Host

1. Train model by tensorflow/caffe/pytorch

```python
def model_forward(inputs):
    c1 = Conv2D(32, (3, 3), activation=None, kernel_initializer='he_normal', padding='same')(inputs)
    c1 = BatchNormalization()(c1)
    c1 = ReLU()(c1)
    c1 = Dropout(0.1)(c1)
    c1 = Conv2D(32, (3, 3), activation=None, kernel_initializer='he_normal', padding='same')(c1)
    c1 = BatchNormalization()(c1)
    c1 = ReLU()(c1)
    p1 = MaxPooling2D((2, 2))(c1)

    c2 = Conv2D(64, (3, 3), activation=None, kernel_initializer='he_normal', padding='same')(p1)
    c2 = BatchNormalization()(c2)
    c2 = ReLU()(c2)
    c2 = Dropout(0.1)(c2)
    c2 = Conv2D(64, (3, 3), activation=None, kernel_initializer='he_normal', padding='same')(c2)
    c2 = BatchNormalization()(c2)
    c2 = ReLU()(c2)
    p2 = MaxPooling2D((2, 2))(c2)

    c3 = Conv2D(128, (3, 3), activation=None, kernel_initializer='he_normal', padding='same')(p2)
    c3 = BatchNormalization()(c3)
    c3 = ReLU()(c3)
    c3 = Dropout(0.2)(c3)
    c3 = Conv2D(128, (3, 3), activation=None, kernel_initializer='he_normal', padding='same')(c3)
    c3 = BatchNormalization()(c3)
    c3 = ReLU()(c3)
    p3 = MaxPooling2D((2, 2))(c3)

    c4 = Conv2D(256, (3, 3), activation=None, kernel_initializer='he_normal', padding='same')(p3)
    c4 = BatchNormalization()(c4)
    c4 = ReLU()(c4)
    return c4
```
On Ubuntu 18.04 Host

2. Quantize model with Vitis AI

```
vai_q_tensorflow quantize \
  --input_frozen_graph \
  ./models/inception_v1_inference.pb \
  --input_nodes input \
  --output_nodes InceptionV1/Logits/Predictions/Reshape_1 \
  --input_fn utils.input_fn_inception_v1_tf \
  --input_shapes ?,224,224,3 \
  --calib_iter 10
```

3. Compile model into FPGA executable file with Vitis AI

```
vai_c_tensorflow \
  -f /PATH/TO/quantize_eval_model.pb -a \ 
  /PATH/TO/arch.json -o /OUTPUTPATH -n netname
```
On ZCU104 FPGA Board

Run .xmodel generated by Vitis AI on FPGA

Prepare input/output tensors on CPU side

Execute DPU runner

Waiting the end of DPU processing