NNDCT user guide

wluo@xilinx.com niuxj@xilinx.com

Table of contents

[*NNDCT user guide* 1](#_Toc23523449)

[Introduction 2](#_Toc23523450)

[Install 2](#_Toc23523451)

[Tool Usage 2](#_Toc23523452)

[Quick Start 3](#_Toc23523453)

[Support op list 5](#_Toc23523454)

# Introduction

Neural Network Deep Compression Toolkit (NNDCT) is designed as a standard platform for neural network deep compression. The architecture is like LLVM—a cross platform compiler for different language. Base on this architecture, working stages of NNDCT follows:

1. Parse neural network computation graph from different frameworks such as Tensorflow, Pytorch and Caffe to intermediate representation (IR).
2. Modify the computation graph based on IR. Pruning, quantization and graph optimization are handled in this stage.
3. Deploy the modified IR on different platform, such as DPU, GPU, CPU, Xilinx AI Engine.
4. Assemble the modified computation graph back to frameworks. In that way abilities and tools in framework such as pre-processing, post processing and distribution system can be used.

Besides being used as platform, NNDCT can also be used as tool chain for deep compression of neural networks, so that full stack solutions for AI applications can be produced. For this target, high-level APIs are given for handling tasks related to neural network deep compression.

For users that need a quick start, there is an example under nndct/example.

Support op list gives supported operation list.

# Install

Follow the README.

# Tool Usage

This chapter introduce using execution tools and APIs to implement quantization, graph optimization procedures. An example can be found under:

nndct/example

For using the APIs and tools, some concepts should be explained.

* **Quantizable graph**: if a graph is quantizable, scanning operations (for configuring quantization) and quantize tensor operations (for quantization) can be automatically generated and applied during graph building stage. And this can be done by setting a quantization FLAG in the environment.
* **Quantization mode**: an integer that indicates which quantization mode the process is using. 0 for turning off quantization. 1 for calibration of quantization. 2 for evaluation of quantized model.

## Quick Start

An example case is nndct/example/resnet18\_quant.py. To call NNDCT, some parts of code needs to be added:

1. Import NNDCT modules

*from nndct\_shared.utils import print\_center\_edge*

*from nndct\_shared.utils import basic\_info*

*from nndct\_shared.utils import check\_diff*

*from nndct\_shared.apis import utils as api\_utils*

*from nndct\_shared.apis.env\_setter import init\_modification\_env*

*from pytorch\_nndct.nndct.apis import register\_modification\_hooks*

*from pytorch\_nndct.nndct.apis import clear\_modification\_hooks*

*from pytorch\_nndct.nndct.apis import load\_state\_dict*

*from pytorch\_nndct.nndct.apis.torch\_modifier import TORCHModifier*

*from pytorch\_nndct.nndct.utils import post\_process*

*from pytorch\_nndct.nndct.load import TORCHStateDictDataLoader*

*from pytorch\_nndct.nndct.models.utils.resnet\_process import load\_data*

1. Integrate NNDCT command line parameter

*args = api\_utils.define\_flags(TORCHModifier, parser)*

1. Set NNDCT options

*init\_modification\_env(*

*TORCHModifier,*

*args,*

*loader\_cls = TORCHStateDictDataLoader,*

*data\_file = file\_path,*

*optimize = args.optimize,*

*quant\_kwargs = {*

*'calibration\_strategy': {'method': 'DiffS'}*

*}*

*)*

1. Transform original models to NNDCT format

*batch\_size = 32*

*# get input data*

*input = torch.randn([batch\_size, 3, 224, 224])*

*model\_gen = api\_utils.rebuilt\_model(*

*model,*

*inputs= (input),*

*state\_dict\_or\_file=file\_path)*

*load\_state\_dict(model\_gen, file\_path)*

1. Register quantization hooks

*register\_modification\_hooks(model\_gen, train=False)*

1. Output and hooks cleanup

*post\_process(model\_gen, args.quant\_file)*

*clear\_modification\_hooks()*

To do calibration of quantization, run the following command line:

*python resnet18\_quant.py --quant\_mode 1*

After it is finished, two important files will be generated under output directory NndctGenData. NndctGen\_Graph.py is transformed NNDCT format model, and NndctGen\_Graph\_quant.json is quanzation steps got.

To do evaluation of quantized model, run the following command line:

*python resnet18\_quant.py --quant\_mode 2*

# Support op list

Limited by resources, NNDCT do not support all operations in pytorch currently. New supported operations will be added from time to time. The table below shows the operation type that NNDCT currently support.

Attention: The list only shows basic supported operations. If the operation is combination of several operations that in the list, the operation is also supported.

|  |  |
| --- | --- |
| Op type | Quantizable |
| torch.nn.Conv2d | Y |
| torch.nn.ConvTranspose2d | Y |
| torch.nn.BatchNorm2d | Y |
| torch.nn.MaxPool2d | Y |
| torch.nn.AvgPool2d | Y |
| torch.nn. AdaptiveAvgPool2d | Y |
| torch.size | N |
| torch.nn.Linear | Y |
| torch.Tensor.view | N |
| torch.cat | Y |
| torch.nn.Dropout | N |
| torch.zeros | N |
| torch.nn.Hardtanh | N |
| torch.transpose | N |
| torch.nn.Softmax | N |
| torch.sum | N |
| torch.mean | Y |
| torch.chunk | N |
| torch.nn.ReLU | N |
| torch.nn.Threshold | N |
| torch.nn.UpsamplingBilinear2d | Y |
| torch.mul | Y |
| torch.add | Y |
| torch.Tensor.contiguous | N |
| torch.floor | N |