Yolov3 Tiny Tutorial: Darknet to Caffe to Xilinx DNNDK

This tutorial is an extension to the Yolov3 Tutorial: Darknet to Caffe to Xilinx DNNDK. The flow of the tutorial is same as described in Edge AI tutorials. Here we mainly focus on the necessary adjustments required to convert Yolov3 Tiny variant.

Download the Yolov3-tiny cfg and weights file.

1. **Directory structure of the Darknet to Caffe project**

```
yolo_convertor
.
├── caffe-master.tar.gz
├── darknet_origin.tar.gz
├── example_yolov3
│   ├── 0_convert.sh
│   ├── 0_model_darknet
│   │   ├── yolov3.cfg
│   ├── 0_test_darknet.sh
│   ├── 1_model_caffe
│   │   └── v3.prototxt
│   ├── 1_test_caffe.sh
│   ├── 2_model_for_quantize
│   │   ├── v3_example.prototxt
│   ├── 2_quantize.sh
│   ├── 3_compile.sh
│   ├── 3_model_after_quantize
│   │   └── deploy.prototxt
│   ├── 4_model_elf
│   │   ├── yolo_kernel_graph.jpg
│   │   └── yolo_kernel.info
│   ├── 5_file_for_test
│   │   ├── calib_data.tar
│   │   ├── calib.txt
│   │   ├── coco.data
│   │   ├── coco.names
│   │   ├── image.txt
│   │   └── test.jpg
│   └── results
├── images
├── README.md
├── yolo_convert.py
└── yolov3_deploy.tar.gz.partaaa
└── yolov3_deploy.tar.gz.partaab
```

Follow the Preparing the Repository step as it is.

Put the downloaded cfg and weights file for yolov3-tiny inside the 0_model_darnet folder.

2. **Edit the yolov3-tiny cfg file. On the line 93,**

Replace this:

```
[maxpool]

size = 2
```

With this:

```
[maxpool]

size = 1
```

Then run the 0_convert.sh file. Before that modify the script file as shown below:

```
$ python ../yolo_convert.py \
       0_model_darknet/yolov3-tiny.cfg        #path to Darknet cfg file \
       0_model_darknet/yolov3-tiny.weights    #path to Darknet weights file \
       1_model_caffe/v3-tiny.prototxt         #path to Caffe prototxt file \
       1_model_caffe/v3-tiny.caffemodel       #path to Caffe caffemodel file
```

This script will convert the Darknet model into two caffe files, v3-tiny.prototxt and v3-tiny.caffemodel and store inside 1_model_caffe folder.

You can test the caffe prototxt using the 1_test_caffe.sh script inside example_yolov3 folder.

3. **Quantize the Caffe Model**

To quantize the Caffe model, copy v3-tiny.prototxt and v3-tiny.caffemodel from 1_model_caffe to the 2_model_for_qunatize. Then modify the v3-tiny.protxt file as shown below:

a. Commenting out the first five lines
b. Adding an ImageData layer with the calibration images for the train phases.

```
name: "Darkent2Caffe"
#####Comment following five lines generated by converter#####
#input: "data"
#input_dim: 1
#input_dim: 3
#input_dim: 416
#input_dim: 416
#####Change input data layer to ImageDate and modify root_folder/source before
run DECENT#####
layer {
name: "data"
type: "ImageData"
top: "data"
top: "label"
include {
  phase: TRAIN
}
transform_param {
  mirror: false
```

```
  yolo_height:416    #change height according to Darknet model
  yolo_width:416     #change width  according to Darknet model
}
image_data_param {
  source:"/PATH_TO/5_file_for_test/calib.txt"         #change path accordingly
  root_folder:"/PATH_TO/5_file_for_test/calib_data/"  #change path accordingly
  batch_size: 1
  shuffle: false
}
}
##### No changes after below layers#####
```

Also make sure to change the source and root folder path accordingly to where you have extracted the 5_file_for_test folder. This contains calibration data.

Edit the 2_quantize.sh script as follows:

```
#Assuming "decent" tool is already in the PATH
$ decent quantize -model 2_model_for_quantize/v3-tiny.prototxt   \
                -weights 2_model_for_quantize/v3-tiny.caffemodel \
                -gpu 0 \
                -sigmoided_layers layer15-conv,layer22-conv  \
                -output_dir 3_model_after_quantize \
                -method 1
```

If you are doing in CPU only mode like inside a virtual machine your script will be the following:

```
#Assuming "decent" tool is already in the PATH
$ decent-cpu quantize -model 2_model_for_quantize/v3-tiny.prototxt   \
                -weights 2_model_for_quantize/v3-tiny.caffemodel \
                -sigmoided_layers layer15-conv,layer22-conv  \
                -output_dir 3_model_after_quantize \
                -method 1
```

Layer15-conv and layer22-conv are the output layers in the Yolov3-tiny as opposed to Yolov3 where layer81-conv, layer93-conv and layer105-conv are the output layers.

4. Compiling the Quantized Model

Modify the deploy.prototxt in the 3_model_after_quantize folder as follows:

```
layer {
name: "data"
type: "Input"
top: "data"
#####Comment following five lines #####
#transform_param {
# mirror: false
# yolo_height: 416
# yolo_width: 416
# }
```

```
#####Nothing change to below layers#####
input_param {
shape {
dim: 1
dim: 3
dim: 416
dim: 416
}
}
}
```

Edit the 3_compile.sh script as follows:

```
#Assume the dnnc-dpu1.3.0 is installed in your $PATH

$ dnnc-dpu1.3.0 --prototxt=3_model_after_quantize/deploy.prototxt \
              --caffemodel=3_model_after_quantize/deploy.caffemodel \
              --dpu=4096FA \
              --cpu_arch=arm64 \
              --output_dir=4_model_elf \
              --net_name=yolo_tiny \
              --mode=normal \
              --save_kernel
```

If you are using DDNDK version 3 which contains dpu 1.4 change the dnnc-dpu1.3.0 to dnnc which will use whatever dpu version you have installed. For Ultra95 change the dpu architecture to 2304FA.

For Ultra96

```
$ dnnc --prototxt=3_model_after_quantize/deploy.prototxt \
              --caffemodel=3_model_after_quantize/deploy.caffemodel \
              --dpu=2304FA \
              --cpu_arch=arm64 \
              --output_dir=4_model_elf \
              --net_name=yolo_tiny \
              --mode=normal \
              --save_kernel
```

This will generate dpu_yolo_tiny.elf file inside the 4_model_elf file. Copy the dpu_yolo_tiny.elf file to the model folder in the yolov3_deploy folder.

Then setup the board and transfer this yolov3_deploy folder to your target board. Ultra96 in our case.

5. Deploying YOLOv3 on the Ultra96 Board

After transferring the yolov3_deploy folder to the board.

Edit the main.cc file which inside yolov3_deploy/src folder. And Make changes as follows:

At Line 239: Change

const vector<string> outputs_node = {"layer81_conv", "layer93_conv", "layer105_conv"};

To

const vector<string> outputs_node = {"layer15_conv", "layer22_conv"};

At Line 395: Change

DPUKernel *kernel = dpuLoadKernel("yolo");

To

DPUKernel *kernel = dpuLoadKernel("yolo_tiny");

Then Save the file. Change the directory to yolov3_deploy and run the make command as follows:

make -j

This will create an executable ***yolo***.

Use the following command to run the executable yolo:

```
#Test image ./yolo coco_test.jpg i
#Test video ./yolo test.video v
```

6. Output of YoloV3-Tiny-Caffe

Final output will be like this:

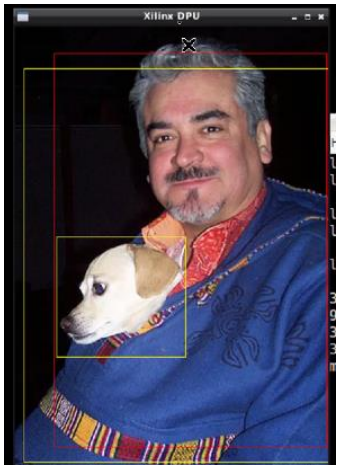

Figure 1: Yolo Implemented on Image for object detection

Figure 2 On video

[We have a demo video of YoloV3 Tiny-Caffe at YouTube: https://youtu.be/AgjMveDPwbo]

***

End of Tutorial!

***