model_quantization_xillinx

December 2, 2020

1 Model quantization

To perform model quantization, we use the Xillinx DNNDK tool (https://www.xilinx.com/support/documentation/user_guides/ug1327-dnndk-user-guide.pdf).

[1]:
```
import os
import sys
sys.path.append(os.path.dirname(os.getcwd()))
```

[2]:
```
import os
import subprocess

import tensorflow as tf
from scripts import evaluate_graph, freeze_model, prepare_data,␣
↪artifacts_reporter, train_model, evaluate_model
```

[3]:
```
DEST_PATH = 'xillinx_model_compilation_results'
DATA_FILE_PATH = os.path.join(os.path.dirname(os.getcwd()), 'datasets/pavia/
↪pavia.npy')
GT_FILE_PAT = os.path.join(os.path.dirname(os.getcwd()), 'datasets/pavia/
↪pavia_gt.npy')
experiment_dest_path = os.path.join(DEST_PATH, 'experiment_0')
data_path = os.path.join(experiment_dest_path, 'data.h5')
os.makedirs(experiment_dest_path, exist_ok=True)
```

2 Prepare the data

To fit into the the pipeline, the data has to be preprocessed. It is achieved by the `prepare_data.main` function. It accepts a path to a `.npy` file with the original cube as well as the corresponding ground truth. In this example, we randomly extract 250 samples from each class (balanced scenario), use 10% of them as validation set, and extract only spectral information of a pixel. The returned object is a dictionary with three keys: `train`, `test` and `val`. Each of them contains an additional dictionary with `data` and `labels` keys, holding corresponding `numpy.ndarray` objects with the data. For more details about the parameters, refer to the documentation of `prepare_data.main` function (located in `scripts/prepare_data`).

[4]:
```
prepare_data.main(data_file_path=DATA_FILE_PATH,
                            ground_truth_path=GT_FILE_PAT,
                            output_path=data_path,
                            train_size=250,
                            val_size=0.1,
                            stratified=True,
                            background_label=0,
                            channels_idx=2,
                            neighborhood_size=None,
                            save_data=True,
                            seed=0)
```

3 Train the model

The function `trian_model.train` executed the trainig procedure. Trained model will be stored under `experiment_dest_path` folder path.

[5]:
```
train_model.train(model_name='model_2d',
                    kernel_size=5,
                    n_kernels=200,
                    n_layers=1,
                    dest_path=experiment_dest_path,
                    data=data_path,
                    sample_size=103,
                    n_classes=9,
                    lr=0.001,
                    batch_size=128,
                    epochs=200,
                    verbose=2,
                    shuffle=True,
                    patience=15,
                    noise=[],
                    noise_sets=[])
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 99, 1, 200)        1200
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 1, 200)        200200
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 1, 200)        200200
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 5, 1, 200)         200200
_________________________________________________________________
flatten (Flatten)            (None, 1000)              0
```

```
_________________________________________________________________
dense (Dense)                (None, 200)               200200
_________________________________________________________________
dense_1 (Dense)              (None, 128)               25728
_________________________________________________________________
dense_2 (Dense)              (None, 9)                 1161
=================================================================
Total params: 828,889
Trainable params: 828,889
Non-trainable params: 0
_________________________________________________________________
Train on 2025 samples, validate on 225 samples
Epoch 1/200
 - 2s - loss: 1.8387 - acc: 0.2889 - val_loss: 1.2870 - val_acc: 0.5156
Epoch 2/200
 - 1s - loss: 1.0412 - acc: 0.5506 - val_loss: 0.9451 - val_acc: 0.5644
Epoch 3/200
 - 1s - loss: 0.8366 - acc: 0.6395 - val_loss: 0.7515 - val_acc: 0.7022
Epoch 4/200
 - 1s - loss: 0.7297 - acc: 0.6706 - val_loss: 0.6807 - val_acc: 0.7022
Epoch 5/200
 - 1s - loss: 0.6336 - acc: 0.7116 - val_loss: 0.5964 - val_acc: 0.7333
Epoch 6/200
 - 1s - loss: 0.5956 - acc: 0.7506 - val_loss: 0.7370 - val_acc: 0.6356
Epoch 7/200
 - 1s - loss: 0.6295 - acc: 0.7077 - val_loss: 0.5713 - val_acc: 0.7422
Epoch 8/200
 - 1s - loss: 0.5567 - acc: 0.7432 - val_loss: 0.6290 - val_acc: 0.7289
Epoch 9/200
 - 1s - loss: 0.5391 - acc: 0.7605 - val_loss: 0.5622 - val_acc: 0.7911
Epoch 10/200
 - 1s - loss: 0.5105 - acc: 0.7689 - val_loss: 0.6307 - val_acc: 0.7022
Epoch 11/200
 - 1s - loss: 0.5392 - acc: 0.7447 - val_loss: 0.5433 - val_acc: 0.7689
Epoch 12/200
 - 1s - loss: 0.5160 - acc: 0.7802 - val_loss: 0.6511 - val_acc: 0.7111
Epoch 13/200
 - 1s - loss: 0.5221 - acc: 0.7620 - val_loss: 0.4940 - val_acc: 0.8133
Epoch 14/200
 - 1s - loss: 0.4687 - acc: 0.8015 - val_loss: 0.5150 - val_acc: 0.7689
Epoch 15/200
 - 1s - loss: 0.4750 - acc: 0.7867 - val_loss: 0.5181 - val_acc: 0.7511
Epoch 16/200
 - 1s - loss: 0.4788 - acc: 0.7926 - val_loss: 0.5191 - val_acc: 0.7644
Epoch 17/200
 - 1s - loss: 0.4350 - acc: 0.8188 - val_loss: 0.4515 - val_acc: 0.7822
Epoch 18/200
 - 1s - loss: 0.4162 - acc: 0.8207 - val_loss: 0.5358 - val_acc: 0.7467
```

```
Epoch 19/200
 - 1s - loss: 0.4717 - acc: 0.8030 - val_loss: 0.4660 - val_acc: 0.7956
Epoch 20/200
 - 1s - loss: 0.3968 - acc: 0.8316 - val_loss: 0.4612 - val_acc: 0.7956
Epoch 21/200
 - 1s - loss: 0.3990 - acc: 0.8380 - val_loss: 0.4333 - val_acc: 0.8133
Epoch 22/200
 - 1s - loss: 0.3856 - acc: 0.8311 - val_loss: 0.4999 - val_acc: 0.7778
Epoch 23/200
 - 1s - loss: 0.4073 - acc: 0.8242 - val_loss: 0.4828 - val_acc: 0.7822
Epoch 24/200
 - 1s - loss: 0.3752 - acc: 0.8356 - val_loss: 0.4472 - val_acc: 0.8133
Epoch 25/200
 - 1s - loss: 0.3650 - acc: 0.8459 - val_loss: 0.4549 - val_acc: 0.8133
Epoch 26/200
 - 1s - loss: 0.3882 - acc: 0.8286 - val_loss: 0.4508 - val_acc: 0.7867
Epoch 27/200
 - 1s - loss: 0.4240 - acc: 0.8119 - val_loss: 0.4096 - val_acc: 0.8533
Epoch 28/200
 - 1s - loss: 0.3642 - acc: 0.8380 - val_loss: 0.4623 - val_acc: 0.8000
Epoch 29/200
 - 1s - loss: 0.3625 - acc: 0.8375 - val_loss: 0.4499 - val_acc: 0.8311
Epoch 30/200
 - 1s - loss: 0.3304 - acc: 0.8543 - val_loss: 0.4298 - val_acc: 0.8133
Epoch 31/200
 - 1s - loss: 0.3398 - acc: 0.8553 - val_loss: 0.4394 - val_acc: 0.7956
Epoch 32/200
 - 1s - loss: 0.3870 - acc: 0.8400 - val_loss: 0.4478 - val_acc: 0.8222
Epoch 33/200
 - 1s - loss: 0.3368 - acc: 0.8568 - val_loss: 0.3683 - val_acc: 0.7822
Epoch 34/200
 - 1s - loss: 0.3120 - acc: 0.8711 - val_loss: 0.3736 - val_acc: 0.8533
Epoch 35/200
 - 1s - loss: 0.3239 - acc: 0.8528 - val_loss: 0.3934 - val_acc: 0.8178
Epoch 36/200
 - 1s - loss: 0.3256 - acc: 0.8642 - val_loss: 0.3959 - val_acc: 0.8267
Epoch 37/200
 - 1s - loss: 0.2845 - acc: 0.8854 - val_loss: 0.3562 - val_acc: 0.8444
Epoch 38/200
 - 1s - loss: 0.2580 - acc: 0.8909 - val_loss: 0.3413 - val_acc: 0.8578
Epoch 39/200
 - 1s - loss: 0.2660 - acc: 0.8928 - val_loss: 0.3595 - val_acc: 0.8533
Epoch 40/200
 - 1s - loss: 0.2936 - acc: 0.8830 - val_loss: 0.3322 - val_acc: 0.8667
Epoch 41/200
 - 1s - loss: 0.2806 - acc: 0.8854 - val_loss: 0.3617 - val_acc: 0.8533
Epoch 42/200
 - 1s - loss: 0.2430 - acc: 0.9017 - val_loss: 0.3646 - val_acc: 0.8622
```

```
Epoch 43/200
 - 1s - loss: 0.2424 - acc: 0.8973 - val_loss: 0.3909 - val_acc: 0.8444
Epoch 44/200
 - 1s - loss: 0.2760 - acc: 0.8840 - val_loss: 0.3287 - val_acc: 0.8489
Epoch 45/200
 - 1s - loss: 0.2437 - acc: 0.8993 - val_loss: 0.3633 - val_acc: 0.8267
Epoch 46/200
 - 1s - loss: 0.3141 - acc: 0.8780 - val_loss: 0.4318 - val_acc: 0.8489
Epoch 47/200
 - 1s - loss: 0.3218 - acc: 0.8711 - val_loss: 0.3201 - val_acc: 0.8889
Epoch 48/200
 - 1s - loss: 0.2497 - acc: 0.8988 - val_loss: 0.3308 - val_acc: 0.8667
Epoch 49/200
 - 1s - loss: 0.2410 - acc: 0.8948 - val_loss: 0.3538 - val_acc: 0.8622
Epoch 50/200
 - 1s - loss: 0.2246 - acc: 0.9086 - val_loss: 0.3172 - val_acc: 0.8578
Epoch 51/200
 - 1s - loss: 0.2185 - acc: 0.9057 - val_loss: 0.3048 - val_acc: 0.8667
Epoch 52/200
 - 1s - loss: 0.2289 - acc: 0.9047 - val_loss: 0.2983 - val_acc: 0.8578
Epoch 53/200
 - 1s - loss: 0.2236 - acc: 0.9081 - val_loss: 0.3055 - val_acc: 0.8444
Epoch 54/200
 - 1s - loss: 0.2205 - acc: 0.9101 - val_loss: 0.3515 - val_acc: 0.8489
Epoch 55/200
 - 1s - loss: 0.2482 - acc: 0.9037 - val_loss: 0.2946 - val_acc: 0.8400
Epoch 56/200
 - 1s - loss: 0.2627 - acc: 0.8904 - val_loss: 0.3426 - val_acc: 0.8622
Epoch 57/200
 - 1s - loss: 0.2434 - acc: 0.8993 - val_loss: 0.3129 - val_acc: 0.8756
Epoch 58/200
 - 1s - loss: 0.2253 - acc: 0.9062 - val_loss: 0.3950 - val_acc: 0.8444
Epoch 59/200
 - 1s - loss: 0.2776 - acc: 0.8889 - val_loss: 0.4312 - val_acc: 0.8533
Epoch 60/200
 - 1s - loss: 0.3172 - acc: 0.8760 - val_loss: 0.3224 - val_acc: 0.8622
Epoch 61/200
 - 1s - loss: 0.2589 - acc: 0.8943 - val_loss: 0.2867 - val_acc: 0.8933
Epoch 62/200
 - 1s - loss: 0.2125 - acc: 0.9151 - val_loss: 0.2847 - val_acc: 0.8933
Epoch 63/200
 - 1s - loss: 0.2174 - acc: 0.9106 - val_loss: 0.2612 - val_acc: 0.8756
Epoch 64/200
 - 1s - loss: 0.2073 - acc: 0.9170 - val_loss: 0.2767 - val_acc: 0.8667
Epoch 65/200
 - 1s - loss: 0.2018 - acc: 0.9141 - val_loss: 0.2956 - val_acc: 0.8889
Epoch 66/200
 - 1s - loss: 0.2112 - acc: 0.9185 - val_loss: 0.3223 - val_acc: 0.8578
```

```
Epoch 67/200
 - 1s - loss: 0.2120 - acc: 0.9151 - val_loss: 0.2848 - val_acc: 0.8667
Epoch 68/200
 - 1s - loss: 0.2122 - acc: 0.9131 - val_loss: 0.2469 - val_acc: 0.8978
Epoch 69/200
 - 1s - loss: 0.2098 - acc: 0.9141 - val_loss: 0.3085 - val_acc: 0.8933
Epoch 70/200
 - 1s - loss: 0.1909 - acc: 0.9230 - val_loss: 0.2961 - val_acc: 0.8800
Epoch 71/200
 - 1s - loss: 0.1933 - acc: 0.9116 - val_loss: 0.3429 - val_acc: 0.8622
Epoch 72/200
 - 1s - loss: 0.2336 - acc: 0.8983 - val_loss: 0.3960 - val_acc: 0.8622
Epoch 73/200
 - 1s - loss: 0.3484 - acc: 0.8672 - val_loss: 0.4104 - val_acc: 0.8178
Epoch 74/200
 - 1s - loss: 0.2864 - acc: 0.8790 - val_loss: 0.2623 - val_acc: 0.8933
Epoch 75/200
 - 1s - loss: 0.2081 - acc: 0.9175 - val_loss: 0.2505 - val_acc: 0.8711
Epoch 76/200
 - 1s - loss: 0.1960 - acc: 0.9146 - val_loss: 0.3222 - val_acc: 0.8578
Epoch 77/200
 - 1s - loss: 0.2106 - acc: 0.9081 - val_loss: 0.2569 - val_acc: 0.8889
Epoch 78/200
 - 1s - loss: 0.2006 - acc: 0.9141 - val_loss: 0.2744 - val_acc: 0.8800
Epoch 79/200
 - 1s - loss: 0.1842 - acc: 0.9244 - val_loss: 0.3155 - val_acc: 0.8756
Epoch 80/200
 - 1s - loss: 0.1955 - acc: 0.9225 - val_loss: 0.2638 - val_acc: 0.8667
Epoch 81/200
 - 1s - loss: 0.1796 - acc: 0.9269 - val_loss: 0.2661 - val_acc: 0.8800
Epoch 82/200
 - 1s - loss: 0.1832 - acc: 0.9289 - val_loss: 0.2981 - val_acc: 0.8844
Epoch 83/200
 - 1s - loss: 0.1810 - acc: 0.9235 - val_loss: 0.2362 - val_acc: 0.9067
Epoch 84/200
 - 1s - loss: 0.2173 - acc: 0.9146 - val_loss: 0.2739 - val_acc: 0.8844
Epoch 85/200
 - 1s - loss: 0.2026 - acc: 0.9111 - val_loss: 0.2930 - val_acc: 0.8711
Epoch 86/200
 - 1s - loss: 0.1748 - acc: 0.9304 - val_loss: 0.2521 - val_acc: 0.9022
Epoch 87/200
 - 1s - loss: 0.1606 - acc: 0.9338 - val_loss: 0.2460 - val_acc: 0.8756
Epoch 88/200
 - 1s - loss: 0.1589 - acc: 0.9348 - val_loss: 0.3433 - val_acc: 0.8889
Epoch 89/200
 - 1s - loss: 0.1688 - acc: 0.9264 - val_loss: 0.2921 - val_acc: 0.8933
Epoch 90/200
 - 1s - loss: 0.1870 - acc: 0.9230 - val_loss: 0.3380 - val_acc: 0.8800
```

```
Epoch 91/200
 - 1s - loss: 0.1732 - acc: 0.9264 - val_loss: 0.2971 - val_acc: 0.8844
Epoch 92/200
 - 1s - loss: 0.2095 - acc: 0.9180 - val_loss: 0.4545 - val_acc: 0.8622
Epoch 93/200
 - 1s - loss: 0.2633 - acc: 0.8968 - val_loss: 0.4939 - val_acc: 0.8222
Epoch 94/200
 - 1s - loss: 0.2562 - acc: 0.8988 - val_loss: 0.2558 - val_acc: 0.8711
Epoch 95/200
 - 1s - loss: 0.1832 - acc: 0.9294 - val_loss: 0.2615 - val_acc: 0.8933
Epoch 96/200
 - 1s - loss: 0.1882 - acc: 0.9235 - val_loss: 0.2606 - val_acc: 0.8933
Epoch 97/200
 - 1s - loss: 0.1748 - acc: 0.9328 - val_loss: 0.2778 - val_acc: 0.8844
Epoch 98/200
 - 1s - loss: 0.1809 - acc: 0.9259 - val_loss: 0.2870 - val_acc: 0.9067
```

4 Evaluate full precision model

Evaluate performance of the model in full precision to later compare to the quantized one.

[ ]:
```
evaluate_model.evaluate(
    model_path=os.path.join(experiment_dest_path, 'model_2d'),
    data=data_path,
    dest_path=experiment_dest_path,
    n_classes=9,
    batch_size=1024,
    noise=[],
    noise_sets=[])
tf.keras.backend.clear_session()
```

5 Freeze model

Freeze the tensorflow model into the `.pb` format.

[7]:
```
freeze_model.main(model_path=os.path.join(experiment_dest_path, 'model_2d'),
                  output_dir=experiment_dest_path)
```

```
INFO:tensorflow:Froze 61 variables.
INFO:tensorflow:Converted 61 variables to const ops.
Frozen model saved at xillinx_model_compilation_results/experiment_0
```

6 Quantize the model

Perform the quantization by running the `quantize.sh` bash script with appropriate parameters. It executes the `decent_q` command from the Xillinx DNNDK library. The output is the `quantize_eval_model.pb` file and a `deploy_model.pb` file, which should be used for compilation for a specific DPU.

[8]:
```
node_names_file = os.path.join(experiment_dest_path,␣
↪'freeze_input_output_node_name.json')
frozen_graph_path = os.path.join(experiment_dest_path, 'frozen_graph.pb')
cmd = '../scripts/quantize.sh ' + node_names_file + ' ' \
        + frozen_graph_path + ' ' + data_path + ' ' + \
        '?,103,1,1' + ' ' + \
        'ml_intuition.data.input_fn.calibrate_2d_input' + ' ' + \
        '128' + ' ' + experiment_dest_path + \
        ' ' + str(0)
f = open(os.path.join(experiment_dest_path, 'call_output.txt'),'w')
env = os.environ.copy()
env['PYTHONPATH'] = os.path.dirname(os.getcwd())
subprocess.call(cmd, shell=True, env=env, stderr=f)
f.close()
```

7 Evaluate the quantized model (graph)

Evaluate the performance of the quantized model to check whether there was any loss in performance. Results for the graph are stored in `inference_graph_metrics.csv`.

[9]:
```
graph_path = os.path.join(experiment_dest_path, 'quantize_eval_model.pb')
evaluate_graph.main(graph_path=graph_path,
                    node_names_path=node_names_file,
                    dataset_path=data_path,
                    batch_size=1024)
tf.keras.backend.clear_session()
```