RetinaFace

Table of Contents

1. Introduction:

This tutorial focuses on porting the model and algorithm of [Retinaface] for inference testing on SOPHON BM1684, BM1684X, and BM1688.

Paper: RetinaFace Paper - "RetinaFace: Single-stage Dense Face Localization in the Wild" proposes a robust single-stage face detector named RetinaFace. It utilizes a combination of extra-supervised and self-supervised multi-task learning to achieve pixel-level localization of faces of different sizes. Specifically, Retinaface makes the following contributions:

(1) Manually annotates 5 facial landmarks (Landmarks) in the WILDER FACE dataset and observes significant improvements in face detection with the help of this additional supervision signal.

(2) Further adds a self-supervised network decoder (mesh decoder) branch to parallelly predict pixel-level 3D shape information of the face alongside existing supervised branches.

(3) On the IJB-C test set, RetinaFace improves the results of state-of-the-art methods (Arcface) in face recognition (FAR=1e6, TAR=85.59%).

(4) With a lightweight backbone network, RetinaFace can run real-time on VGA resolution images on a single processor.

Dataset

The dataset used in the Retinaface paper is the WIDER FACE dataset, consisting of 32,203 images and 393,703 face bounding boxes. The WIDER FACE dataset is divided into training (40%), validation (10%), and testing (50%) subsets by randomly sampling from 61 scene categories. WIDER FACE dataset download link: http://shuoyang1213.me/WIDERFACE/

Model and Data Preparation

You need to prepare the dataset for testing.

In the scripts directory, this tutorial provides a download script (download.sh) for relevant models and datasets. You can also prepare the models and datasets yourself and refer to Section 4. Model Conversion for model conversion.

```
chmod -R +x scripts/
./scripts/download.sh
```

After execution, the models will be saved to `data/models`, the image dataset will be downloaded and extracted to `data/images/`, and the videos will be saved to `data/videos/`.

The downloaded models include:

onnx/retinaface_mobilenet0.25.onnx: Original model
BM1684/retinaface_mobilenet0.25_fp32_1b.bmodel: FP32 BModel for BM1684, batch_size=1
BM1684/retinaface_mobilenet0.25_int8_1b.bmodel: INT8 BModel for BM1684, batch_size=1
BM1684/retinaface_mobilenet0.25_int8_4b.bmodel: INT8 BModel for BM1684, batch_size=4
BM1684X/retinaface_mobilenet0.25_fp32_1b.bmodel: FP32 BModel for BM1684X, batch_size=1
BM1684X/retinaface_mobilenet0.25_fp16_1b.bmodel: FP16 BModel for BM1684X, batch_size=1
BM1684X/retinaface_mobilenet0.25_int8_1b.bmodel: INT8 BModel for BM1684X, batch_size=1
BM1684X/retinaface_mobilenet0.25_int8_4b.bmodel: INT8 BModel for BM1684X, batch_size=4
BM1688/retinaface_mobilenet0.25_fp32_1b.bmodel: FP32 BModel for BM1688, batch_size=1
BM1688/retinaface_mobilenet0.25_fp16_1b.bmodel: FP16 BModel for BM1688, batch_size=1
BM1688/retinaface_mobilenet0.25_int8_1b.bmodel: INT8 BModel for BM1688, batch_size=1
BM1688/retinaface_mobilenet0.25_int8_4b.bmodel: INT8 BModel for BM1688, batch_size=4
BM1688/retinaface_mobilenet0.25_fp32_1b_2core.bmodel: FP32 Dual-core BModel for BM1688, batch_size=1
BM1688/retinaface_mobilenet0.25_fp16_1b_2core.bmodel: FP16 Dual-core BModel for BM1688, batch_size=1
BM1688/retinaface_mobilenet0.25_int8_1b_2core.bmodel: INT8 Dual-core BModel for BM1688, batch_size=1
BM1688/retinaface_mobilenet0.25_int8_4b_2core.bmodel: INT8 Dual-core BModel for BM1688, batch_size=4
The downloaded data includes:

WIDERVAL: Testing set
face01.jpg to face05.jpg: Test images
station.avi: Test video

Model Information:

Original Model: retinaface_mobilenet0.25.onnx
Overview: Face detection model

Backbone Network: MobileNet0.25 ResNet50
Training Dataset: WiderFace
Input Data: [batch_size, 3, 640, 640], FP32, NCHW
Output Data: [batch_size, 68, 18], FP32
Preprocessing: Resize, subtract mean, divide variance, HWC to CHW
Postprocessing: Filter NMS
Please note that the version of ONNX used for this model is 1.6.0. If the ONNX version in your environment is too high, it may result in compilation failures.

Model Conversion

The exported models need to be compiled into BModels to run on the SOPHON TPU. If you are using the downloaded BModels, you can skip this section. It is recommended to use TPU-MLIR for model compilation.
Before compiling the model, TPU-MLIR needs to be installed. For specific instructions, refer to the TPU-MLIR environment setup. Once installed, navigate to the tutorial directory within the TPU-MLIR environment. Use TPU-MLIR to compile the ONNX model into a BModel. For detailed instructions, refer to the "3. Compile ONNX Models" section in the "TPU-MLIR Quick Start Guide" (please obtain the SDK from the official website of Alchemy, the manufacturer of SOPHON TPU).

Generate FP32 Bmodel

This tutorial provides a script in the scripts directory for compiling FP32 BModel using TPU-MLIR. Please make sure to modify the paths to the ONNX model, the output model directory, input sizes (shapes), etc., in the gen_fp32bmodel_mlir.sh script. When executing the script, specify the target platform for BModel execution (supporting BM1684/BM1684X/BM1688), for example:

```
./scripts/gen_fp32bmodel_mlir.sh bm1684
#or
./scripts/gen_fp32bmodel_mlir.sh bm1684x
#or
./scripts/gen_fp32bmodel_mlir.sh bm1688
```

Generate FP16 Bmodel

This tutorial provides a script in the scripts directory for compiling FP16 BModel using TPU-MLIR. Please make sure to modify the paths to the ONNX model, the output model directory, input sizes (shapes), etc., in the gen_fp16bmodel_mlir.sh script. When executing the script, specify the target platform for BModel execution (supporting BM1684X/BM1688), for example:

```
./scripts/gen_fp16bmodel_mlir.sh bm1684x
#or
./scripts/gen_fp16bmodel_mlir.sh bm1688
```

Executing the above command will generate the retinaface_mobilenet0.25_fp16_1b.bmodel file in data/models/BM1684X/ or data/models/BM1688/, which is the converted FP16 BModel.

Generate INT8 Bmodel

This tutorial provides two scripts in the scripts directory for quantizing INT8 BModels. Please make sure to modify the paths to the ONNX model, the output model directory, input sizes (shapes), etc., in the gen_int8bmodel_mlir_qtable.sh and gen_int8bmodel_mlir_sensitive_layer.sh scripts. When executing the scripts, input the target platform for BModel execution (supporting BM1684/BM1684X/BM1688), for example:

```
./scripts/gen_int8bmodel_mlir_qtable.sh bm1684
./scripts/gen_int8bmodel_mlir_sensitive_layer.sh bm1684
#或
./scripts/gen_int8bmodel_mlir_qtable.sh bm1684x
./scripts/gen_int8bmodel_mlir_sensitive_layer.sh bm1684x
#或
#BM1688 暂不支持 sensitive_layer
./scripts/gen_int8bmodel_mlir_qtable.sh bm1688
```

In this tutorial, mixed precision is used for quantizing INT8 BModels. You can further improve model accuracy by modifying the --max_float_layers parameter in gen_int8bmodel_mlir_sensitive_layer.sh or adjusting the head parameter in gen_int8bmodel_mlir_qtable.sh to increase the number of floating-point layers used.

The above scripts will generate files such as retinaface_mobilenet0.25_int8_1b.bmodel and retinaface_mobilenet0.25_int8_4b.bmodel in data/models/BM1684/, data/models/BM1684X/, or data/models/BM1688/, which are the converted INT8 Bmodels.

Example Testing

C++ Example
Python Example

Accuracy Testing

6.1 Testing Method
You may need to install other third-party libraries:

```
$ pip3 install -r python/requirements.txt
```

This tutorial provides accuracy testing tools in the tools directory. These tools can compare the predicted results on the WIDERFACE test set with the ground truth to calculate the accuracy of face detection. The specific testing command is as follows:

```
cd tools/widerface_evaluate
tar -zxvf widerface_txt.tar.gz
# 请根据实际情况，将第 5 节生成的包含预测结果 txt 文件的路径填入 transfer.py 中的变量 source_txt, 并保证 widerface_txt/的二级目录为空
python3 transfer.py
python3 setup.py build_ext --inplace
python3 evaluation.py
```

After execution, the accuracy on the WIDERFACE easy test set will be printed.

6.2 Testing Results

In the Pytorch_Retinaface, the accuracy on the WIDERFACE easy test set with the original image scale is 90.7%. In this tutorial, the resize strategy is changed, and images are resized to 640*640 for inference, resulting in an accuracy of 89.5% on this test set.

On different testing platforms, using different examples and models to test on data/images/WIDERVAL with conf_thresh=0.02, nms_thresh=0.4, the Retinaface accuracy testing results are as follows:

Performance Testing

7.1 bmrt_test

You can use bmrt_test to test the theoretical performance of the model:

```
bmrt_test --bmodel {path_of_bmodel}
```

You can also refer to Section 5. Example Testing to print the actual performance metrics during program execution.
There may be some fluctuations in performance metrics during testing, which is normal.

The "calculate time" in the test results represents the inference time of the model. For models with multiple batch sizes, the time should be divided by the corresponding batch size to get the theoretical inference time per image. The theoretical inference time for various models is as follows:

Testing Explanation:

infer_time: Actual inference time per image during program execution.
QPS: Number of images processed per second by the program.
The results of performance testing may exhibit some fluctuations.