# Auto Tuning Tool

# 1 Introduction

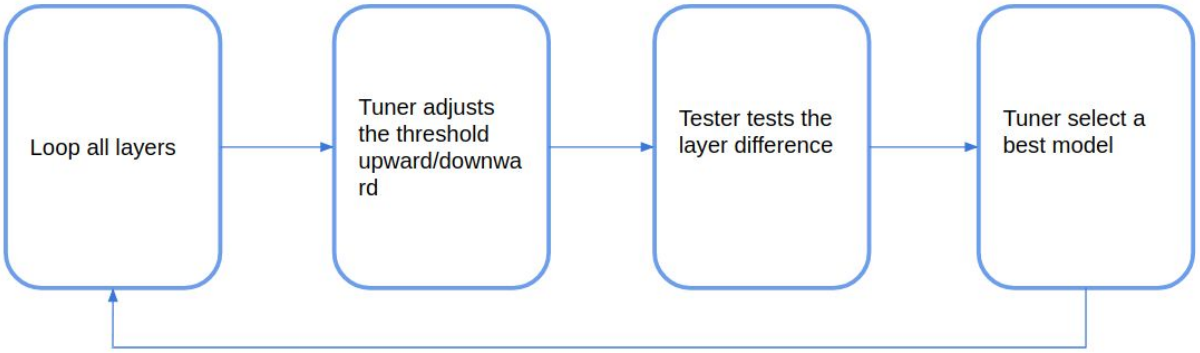
This tool is to help you adjust the threshold of the calibration table to get a more accurate model. We analyse the feature difference between floating point and fixed point model, and try each layer's threshold upward and downward to see if we can find a better threshold or not. Iterate all net and output the final tuned model.

# 2 Preparation

The same as calibration tool.

# 3 How to Run

## 3.1 Flow chart




## 3.2 Scripts

The main.sh is the entrance of the auto tuning tool. It calls main.py to run the auto tune procedure. The parameters are as below:

| Parameter | Description |
|---|---|
| model | The floating point caffe model |
| proto | The caffe prototxt |

| | |
|---|---|
| calibration_model | The original int8 caffe model to be tuned |
| calibration_proto | The calibration table pb2 |
| output_path | The temperary path to store the tune models |
| data_list | The inference data list. Please refer to *3.2 ingerence* for more detail |
| data_limit | The inference data limit number. Please refer to *3.2 Inference* for more detail |
| image_params | The image preprocess parameter json file path. Please refer to *3.3 Image Preprocess* for more detail |
| ignore_layer_list | Ignore layers that you don't want to tune. |

The main.sh contains two parts, *test* and *tune*. Each part has it own script called test.sh and tune.sh. You can run them separately. This script will loop all layers in your model and find a better quantization threshold for each layer.

The test.sh will generate a report.log and record the int8/fp32 difference of one target layer. The *int8_layer* parameter of test.sh is used to specify the target layer. If empty string, the script will generate the difference of the model.

The tune.sh is to adjust the threshold of a target layer. The *tune_layer* parameter is to specify the target layer and *tune_diff* is the difference of the target layer before tuning. The script will generate all tuned model in the *output_path*.

## 3.3 Inference Data

The inference data is used to generate the feature difference of floating and fixed point. This tool will search all images defined in the *data_list* param. You can also define *data_limit* to limit the search number. The data should cover a variety of different scenes to approach the real world input. In our experience, 5 to 10 numbers of data is enough for testing.

The data, like calibration tool, should have the same preprocess as when testing or the feature difference can not fully reflect the real world situation.

## 3.4 Image Preprocess

The preprocess is writing in the *prepare_image* function in test_utils.py. We provide a general template but you can modify to your own preprocess as well. We

use a json file to pass the parameters to the function. Theres is an example file named image_params.json in the release folder and below are the parameters:

| Parameter | Description |
|---|---|
| size | The target image size([h w]) |
| padding | Specify padding when resize or not |
| mirror | The mirror method. Using the traditional opencv flip.<br>1: horizontal flip<br>0: vertical flip<br>-1: horizontal and vertical flip<br>Ignore when other value. |
| color_format | The color format order. Only support ‘rgb’ or ‘bgr’ |
| r/g/b_mean | The mean value of r/g/b channel. |
| scale | The target image scale |

The image preprocess is strictly affect the final result. Make sure that using the right procedure.

## 3.5 Feature Difference

The difference metric can affect the final result as well. We suggest that the metric be as similar as the original accuracy metric so the difference between fp32 and int8 feature can reflect the accuracy difference the most. We provide two different metric in the *evaluation_utils.py*. One is a simple square distance which in our experience has some advantage for several applications like image classification, face alignment. The second is a metric specific to yolo, which would transform the feature map to box, class, score pair then calculate the square distance of this pair.