零基础入门旷视天元MegEngine

模型优化

讲师：王鹏

欢迎加入“天元开发者交流群”

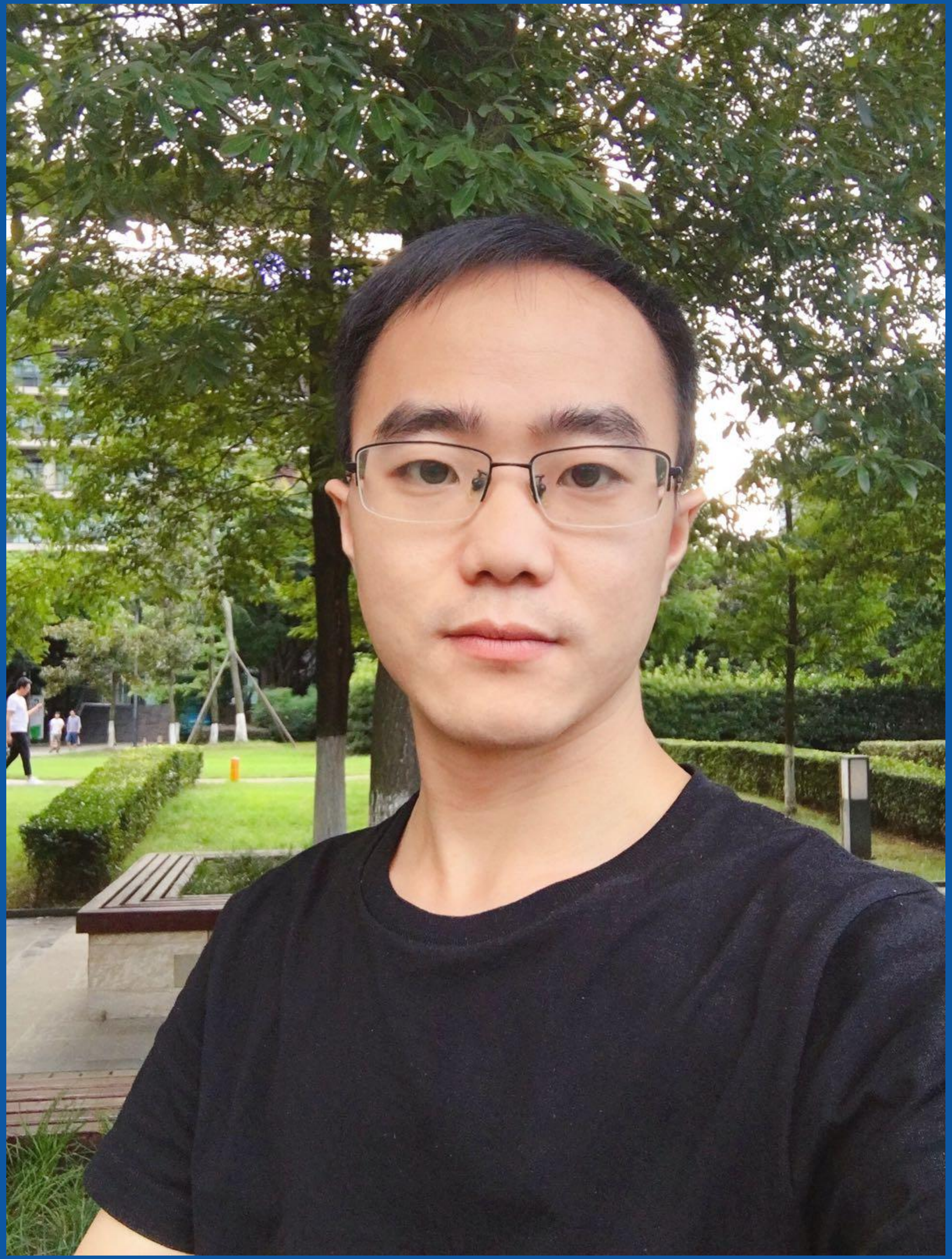
天元开发者交流群

群号：1029741705

扫一扫二维码，加入群聊。

讲师介绍：

多摄算法研究员，方向为多视图三维重建与深度学习融合，参与实时预览虚化，光学变焦，超广角畸变，单目深度估计等多个项目的研发与落地，申请相关专利 8 篇。



课程大纲：

介绍在MegEngine框架下的模型优化

- 概述
- 量化方法介绍
- 实例讲解

MEGVII 旷视

1 概述

2 量化方法介绍

3 实例讲解

天元开发者交流群

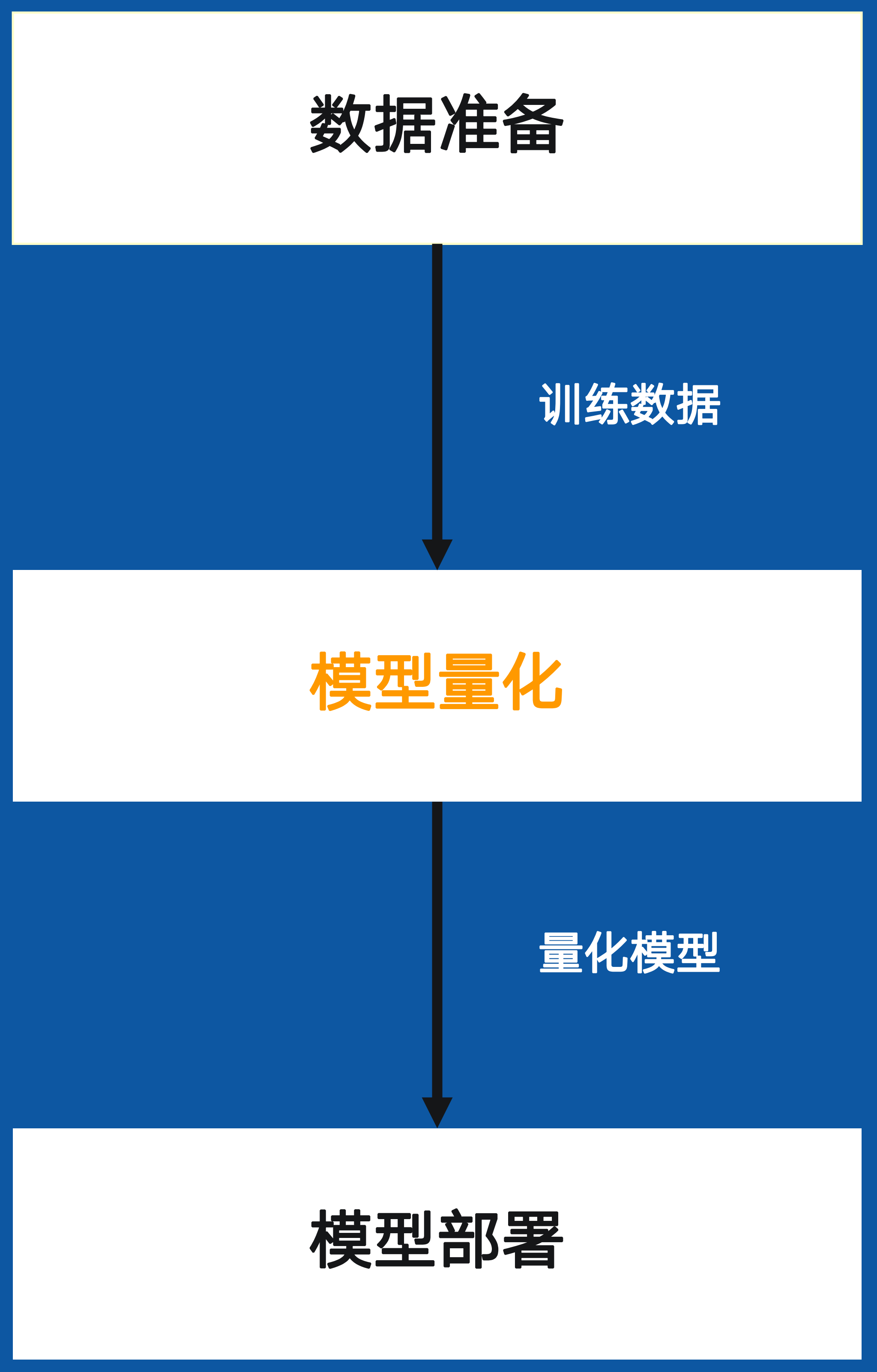
群号：1029741705

扫一扫二维码，加入群聊。

概述

定义

模型量化是指对模型中的 Weight 和 Activations 进行量化，将训练模型时常用的 32 位浮点数 （ FP32 ） 转化为更少位数的数值类型（如 INT 8 等）来进行计算和存储的技术。



模型部署示意图

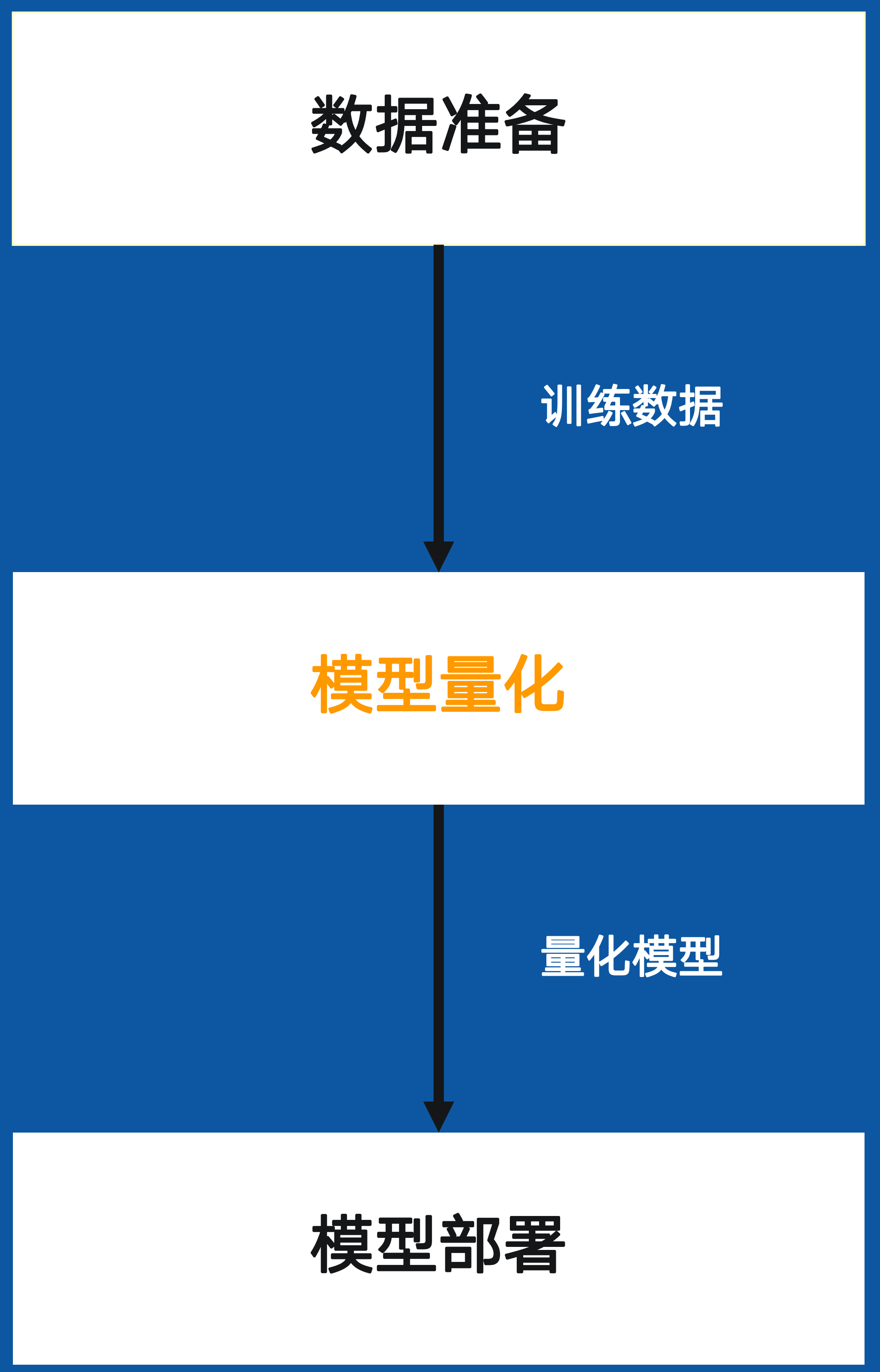
概述

定义

模型量化是指对模型中的 Weight 和 Activations 进行量化，将训练模型时常用的 32 位浮点数 （ FP32 ） 转化为更少位数的数值类型（如 INT 8 等）来进行计算和存储的技术。

作用

- 降本： 降低模型存储空间
- 增效： 加快模型推理 （Inference）速度
- 节约： 减少运算设备访存占用



模型部署示意图

概述

MEGVII 旷视

定义

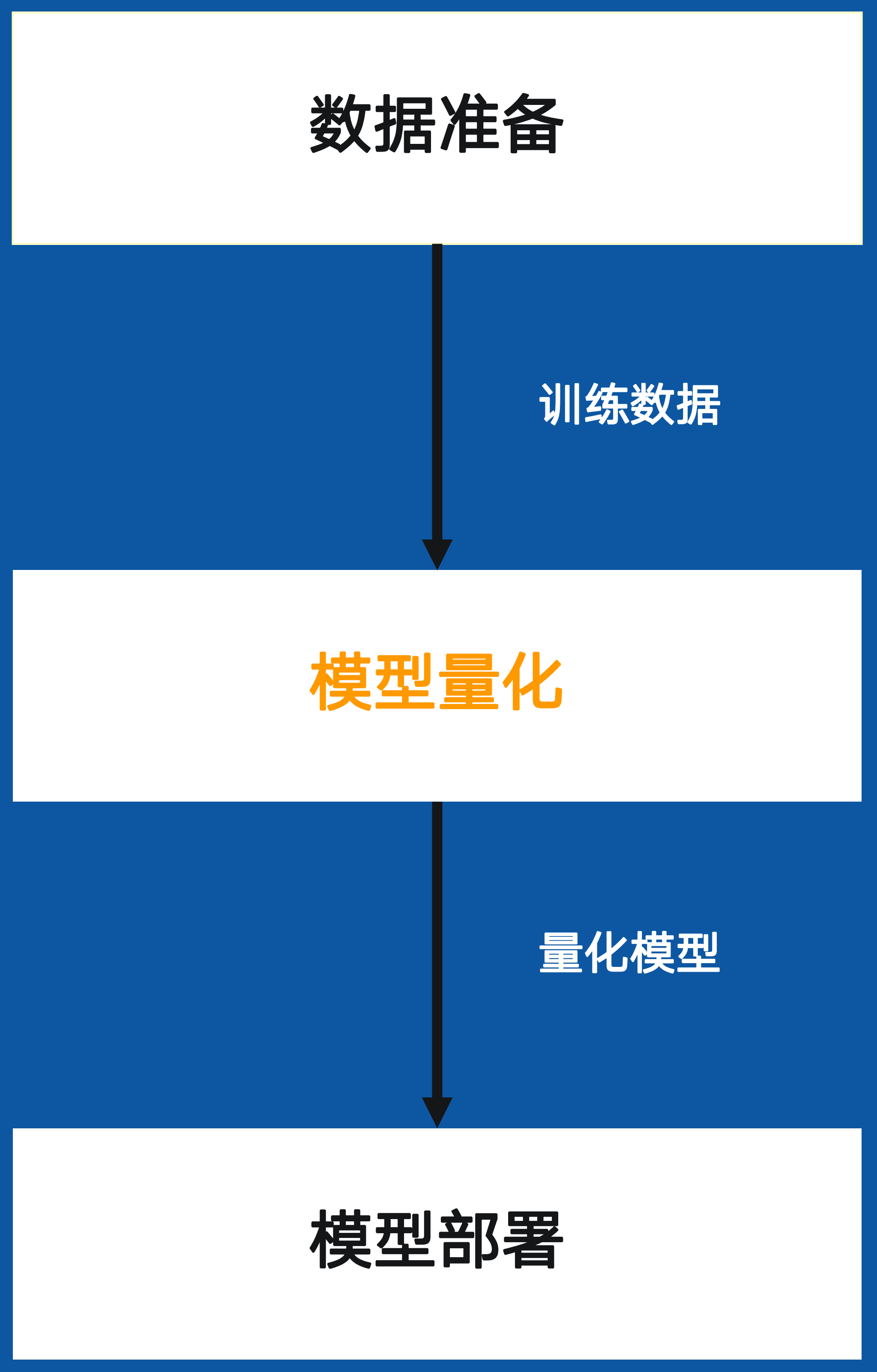
模型量化是指对模型中的 Weight 和 Activations 进行量化，将训练模型时常用的 32 位浮点数 （ FP32 ） 转化为更少位数的数值类型（如 INT 8 等）来进行计算和存储的技术。

作用

- 降本： 降低模型存储空间
- 增效： 加快模型推理 （Inference）速度
- 节约： 减少运算设备访存占用

核心

如何减小量化中信息损失对模型的负面影响？



模型部署示意图

概述

定义

模型量化是指对模型中的 Weight 和 Activations 进行量化，将训练模型时常用的 32 位浮点数 （ FP32 ） 转化为更少位数的数值类型（如 INT 8 等）来进行计算和存储的技术。

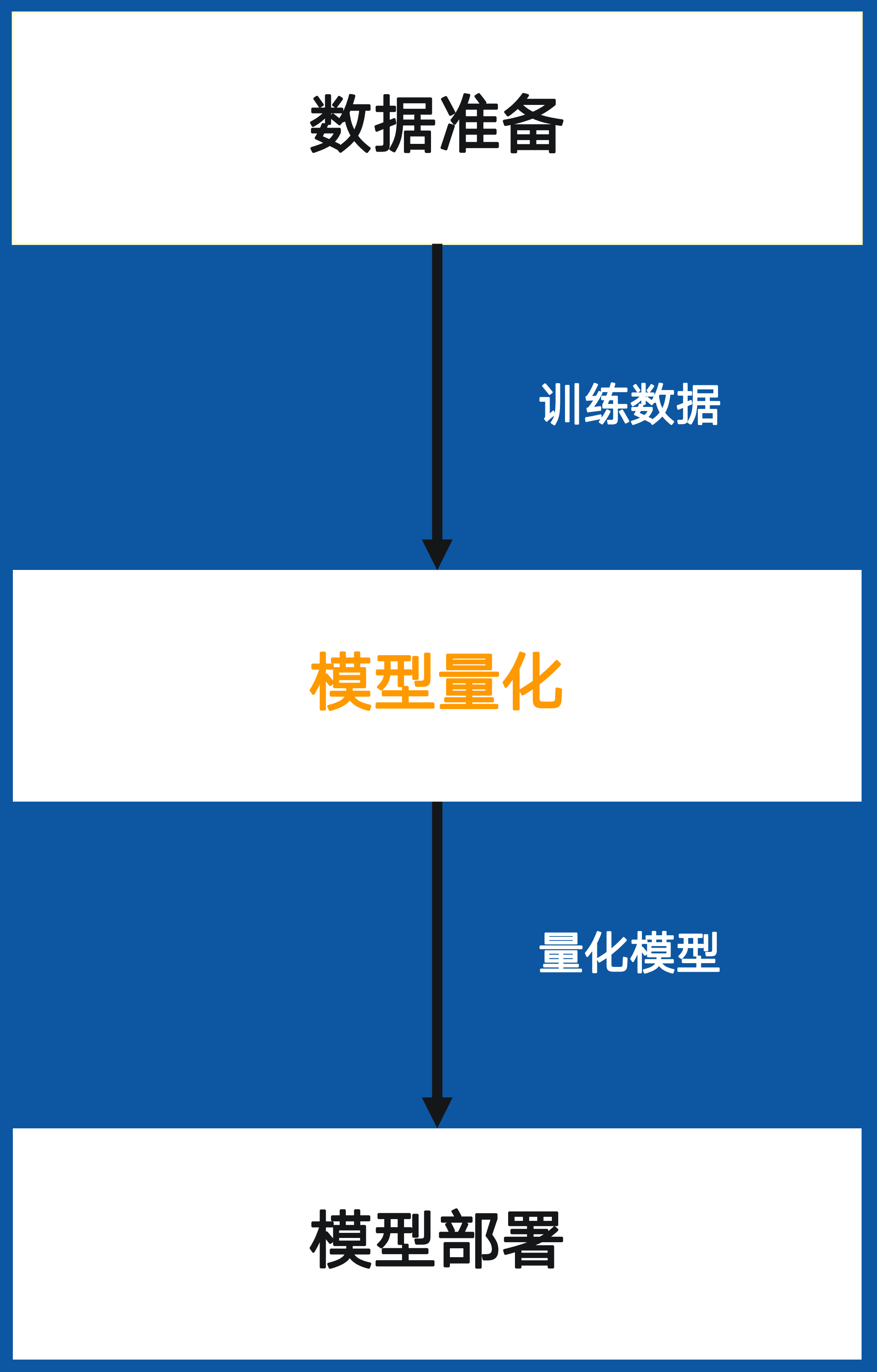
作用

- 降本： 降低模型存储空间
- 增效： 加快模型推理 （Inference）速度
- 节约： 减少运算设备访存占用

核心

如何减小量化中信息损失对模型的负面影响？

MegEngine使用了一系列精细量化处理，使其掉点可以变得微乎其微，并能支持正常的部署使用



模型部署示意图

概述

定义

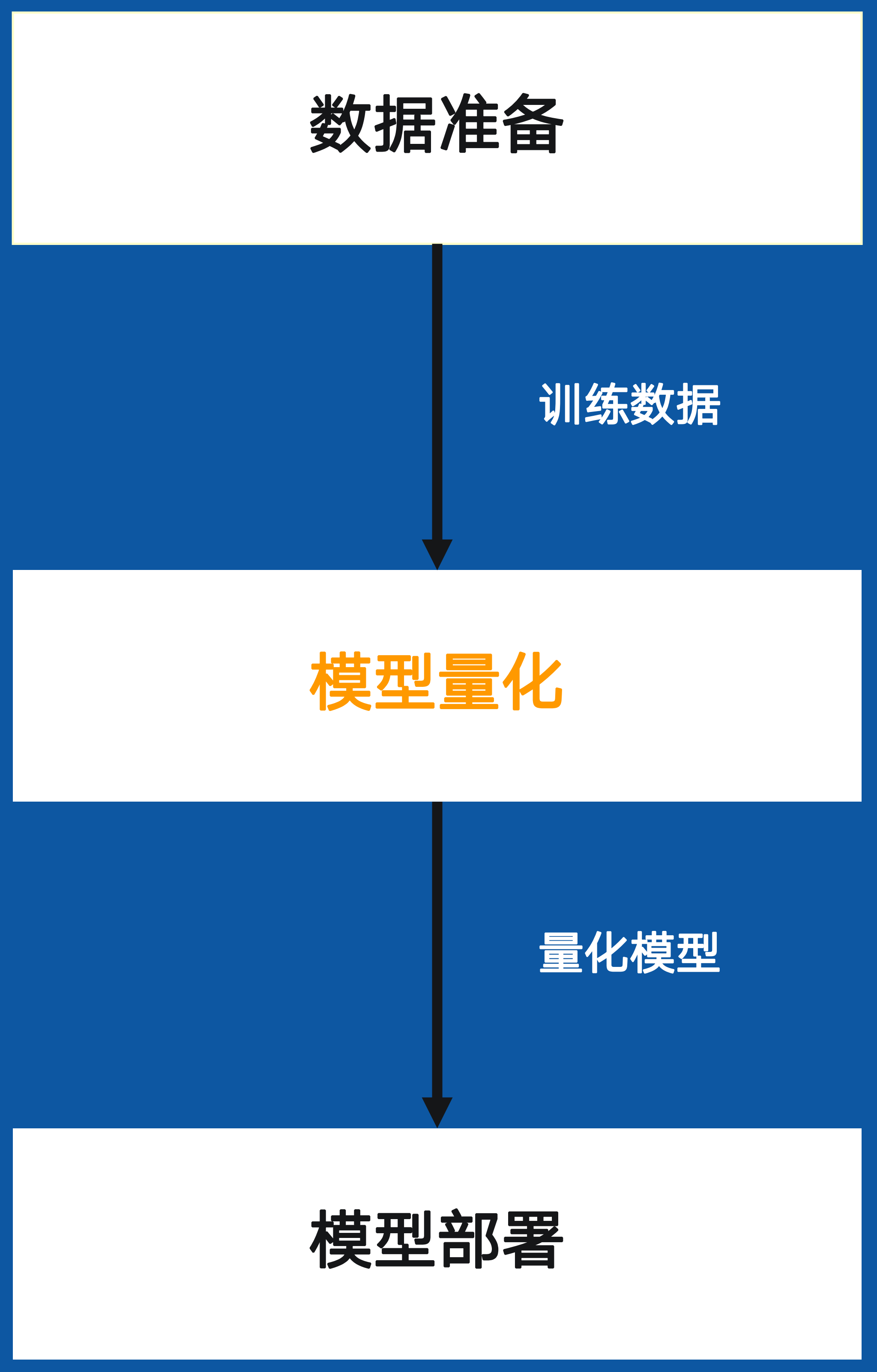
模型量化是指对模型中的 Weight 和 Activations 进行量化，将训练模型时常用的 32 位浮点数 （ FP32 ） 转化为更少位数的数值类型（如 INT 8 等）来进行计算和存储的技术。

作用

- 降本： 降低模型存储空间
- 增效： 加快模型推理 （Inference）速度
- 节约： 减少运算设备访存占用

核心

MegEngine框架基于ImageNet数据的量化结果对比			
Resnet18	FP32	INT8	(FP32 -INT8) * 100 / FP32
Acc	69.824	69.754	0.1003
Speed (ms)	95.4	61.5	35.5346
Size (MB)	46.804	13.248	71.6933



模型部署示意图

MEGVII 旷视

1 概述

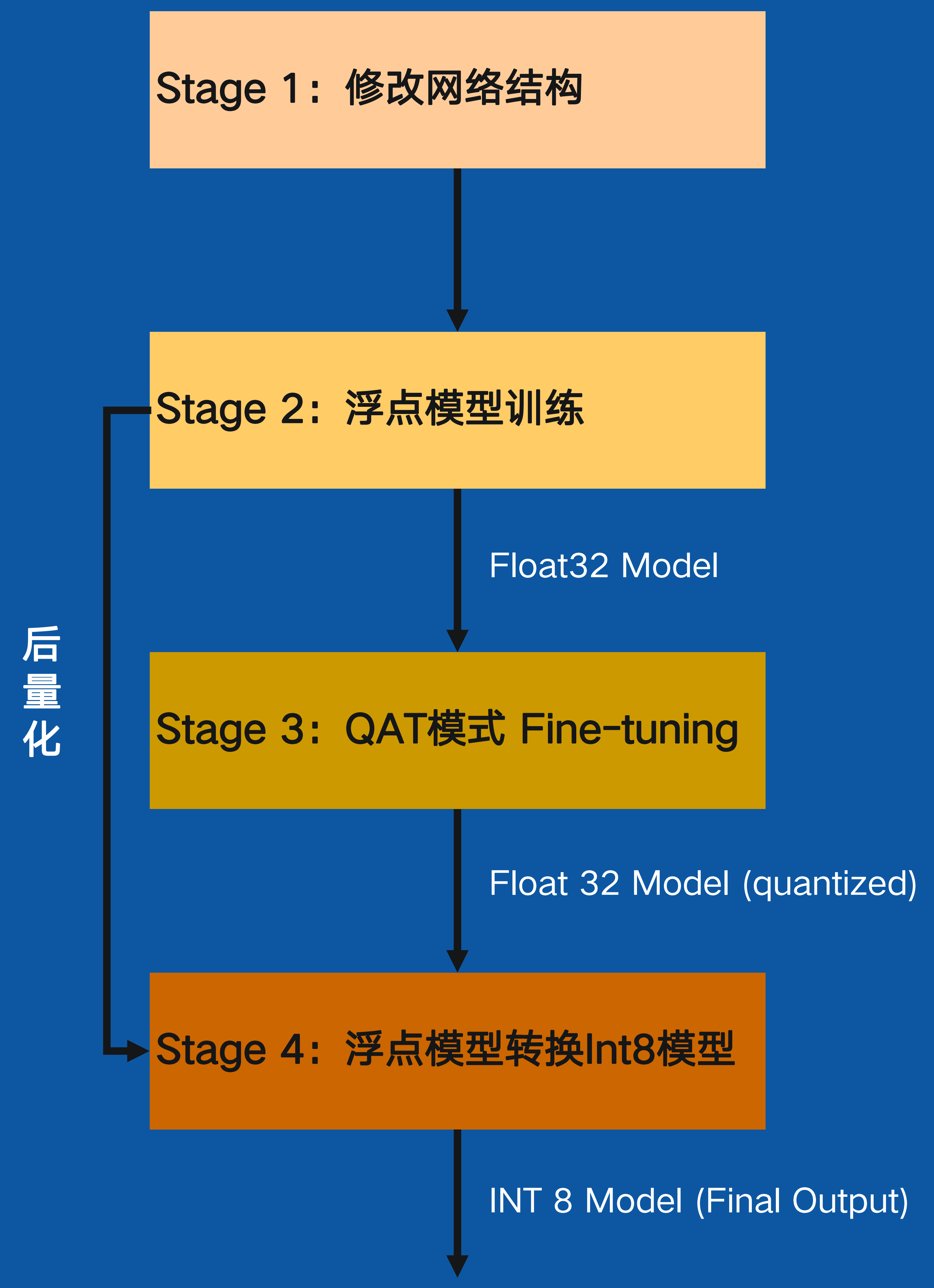
2 量化方法介绍

3 实例讲解

量化方法介绍

量化方法根据介入时机不同大致可以分为：

- 量化感知训练（Quantization Aware Training, QAT）
- 后量化（Post Quantization）



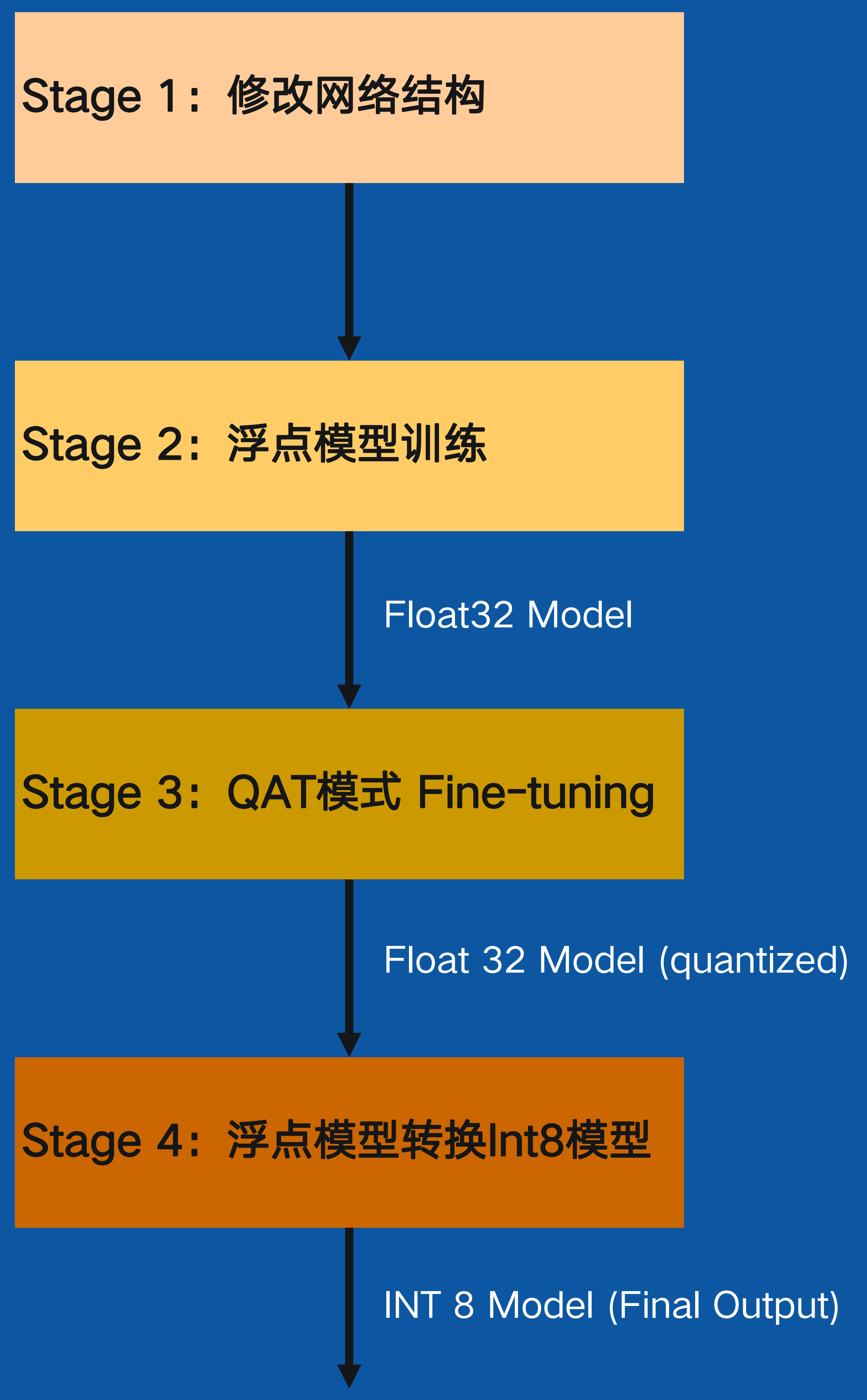
模型量化示意图

量化方法介绍

量化方法根据介入时机不同大致可以分为：

- 量化感知训练（Quantization Aware Training, QAT）

QAT 是指有训练数据的情况下，在原网络中插入一些 FakeQuantize 算子，模拟Weight和Activations被截断后精度降低的情形，并在此基础上进行 Fine - tuning 训练，使得模型能够提前感知精度降低的操作。



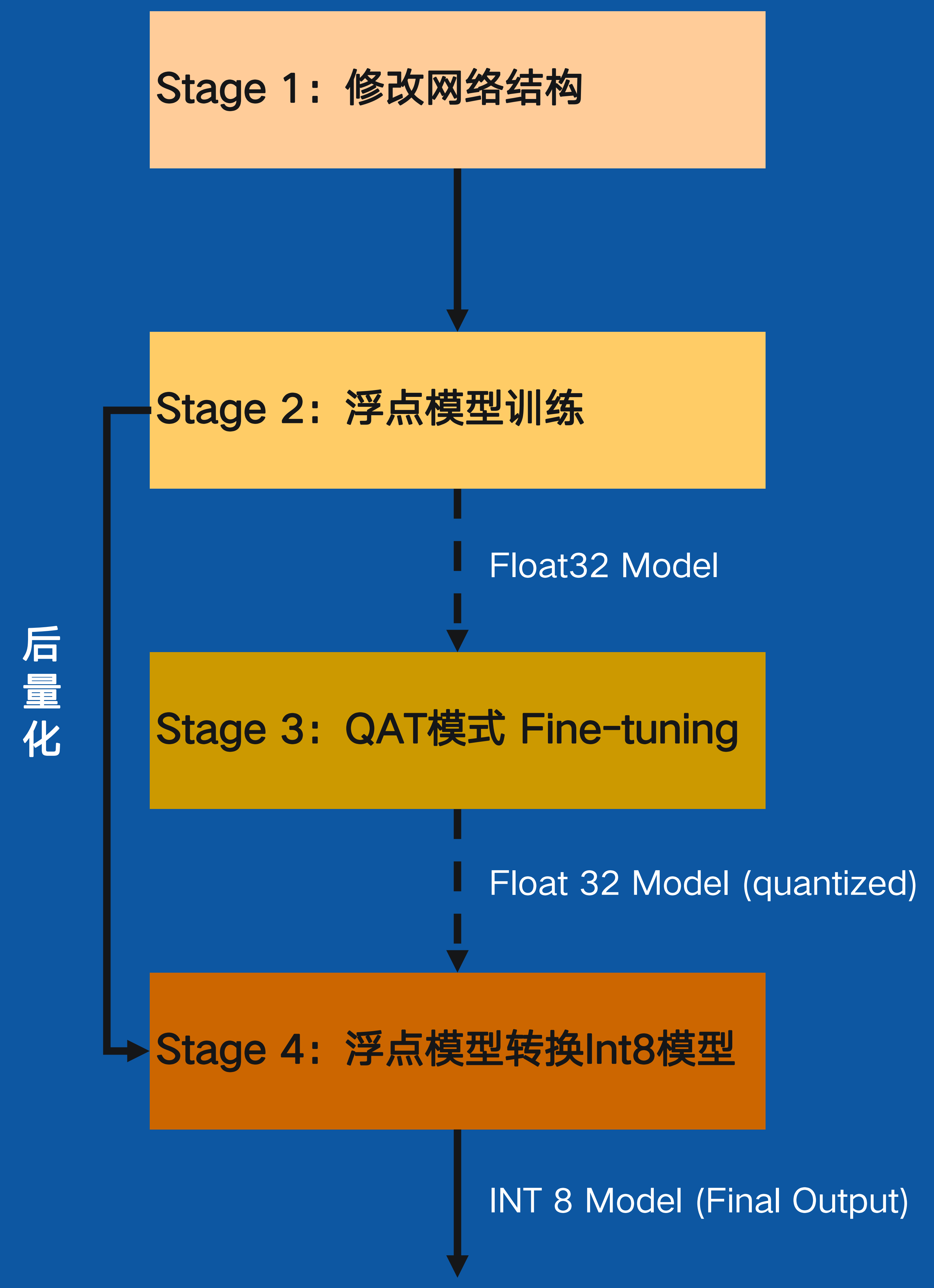
模型量化示意图

量化方法介绍

量化方法根据介入时机不同大致可以分为：

- 后量化（Post Quantization）

后量化不进行网络模型的 Fine - tuning，通过一些标定数据来统计网络中Weight 和 Activations 的数值范围，然后直接进行数值转换。



模型量化示意图

量化方法介绍

MEGVII 旷视

量化

QConfig | Fused Module

量化方法介绍

MEGVII 旷视

量化

QConfig	Fused Module

class megengine.quantization.qconfig.**QConfig**(*act_observer*, *weight_observer*, *fake_quant*)

Bases: object

A config class indicating how to do quantize toward QATModule's activation and weight. See set_qco

Parameters:

- **weight_observer** – interface to instantiate an Observer indicating how to collect s
- **act_observer** – similar to weight_observer but toward activation.
- **fake_quant** – interface to instantiate a FakeQuantize indicating how to do fake_qu
  target tensor, for better control on enable and disable.

量化方法介绍

MEGVII 旷视

量化

QConfig | Fused Module

class megengine.quantization.qconfig.**QConfig**(*act_observer*, *weight_observer*, *fake_quant*)

Bases: object

A config class indicating how to do quantize toward QATModule's activation and weight. See set_qco

Parameters:
- **weight_observer** – interface to instantiate an Observer indicating how to collect sc
- **act_observer** – similar to weight_observer but toward activation.
- **fake_quant** – interface to instantiate a FakeQuantize indicating how to do fake_qu target tensor, for better control on enable and disable.

```
# Default EMA QConfig for QAT.
ema_fakequant_qconfig = QConfig(
    weight_observer=MinMaxObserver,
    act_observer=ExponentialMovingAverageObserver,
    fake_quant=FakeQuantize,
)
```

天元开发者交流群

群号：1029741705

扫一扫二维码，加入群聊。

量化方法介绍

量化

QConfig

class megengine.quantization.qconfig.**QConfig**(*act_observer*, *weight_observer*, *fake_quant*)

Bases: object

A config class indicating how to do quantize toward QATModule's activation and weight. See set_qco

Parameters:
- **weight_observer** – interface to instantiate an Observer indicating how to collect sc
- **act_observer** – similar to weight_observer but toward activation.
- **fake_quant** – interface to instantiate a FakeQuantize indicating how to do fake_qu target tensor, for better control on enable and disable.

```
# Default EMA QConfig for QAT.
ema_fakequant_qconfig = QConfig(
    weight_observer=MinMaxObserver,
    act_observer=ExponentialMovingAverageObserver,
    fake_quant=FakeQuantize,
)
```

Fused Module

megengine.module package

megengine.module.qat package

megengine.module.quantized package

量化方法介绍

量化

QConfig

Fused Module

Observer

FakeQuantize

Module

QAT Module

Quantized Module

量化方法介绍

MEGVII 旷视

量化

QConfig

Fused Module

Observer

FakeQuantize

Module

QAT Module

Quantized Module

class megengine.quantization.fake_quant.**FakeQuantize**(*dtype*, *enable=True*)

Bases: megengine.module.module.Module

A module to do quant and dequant according to observer's scale and zero_point.

天元开发者交流群

群号：1029741705

扫一扫二维码，加入群聊。

量化方法介绍

量化

QConfig | Fused Module

Observer | FakeQuantize | Module | QAT Module | Quantized Module

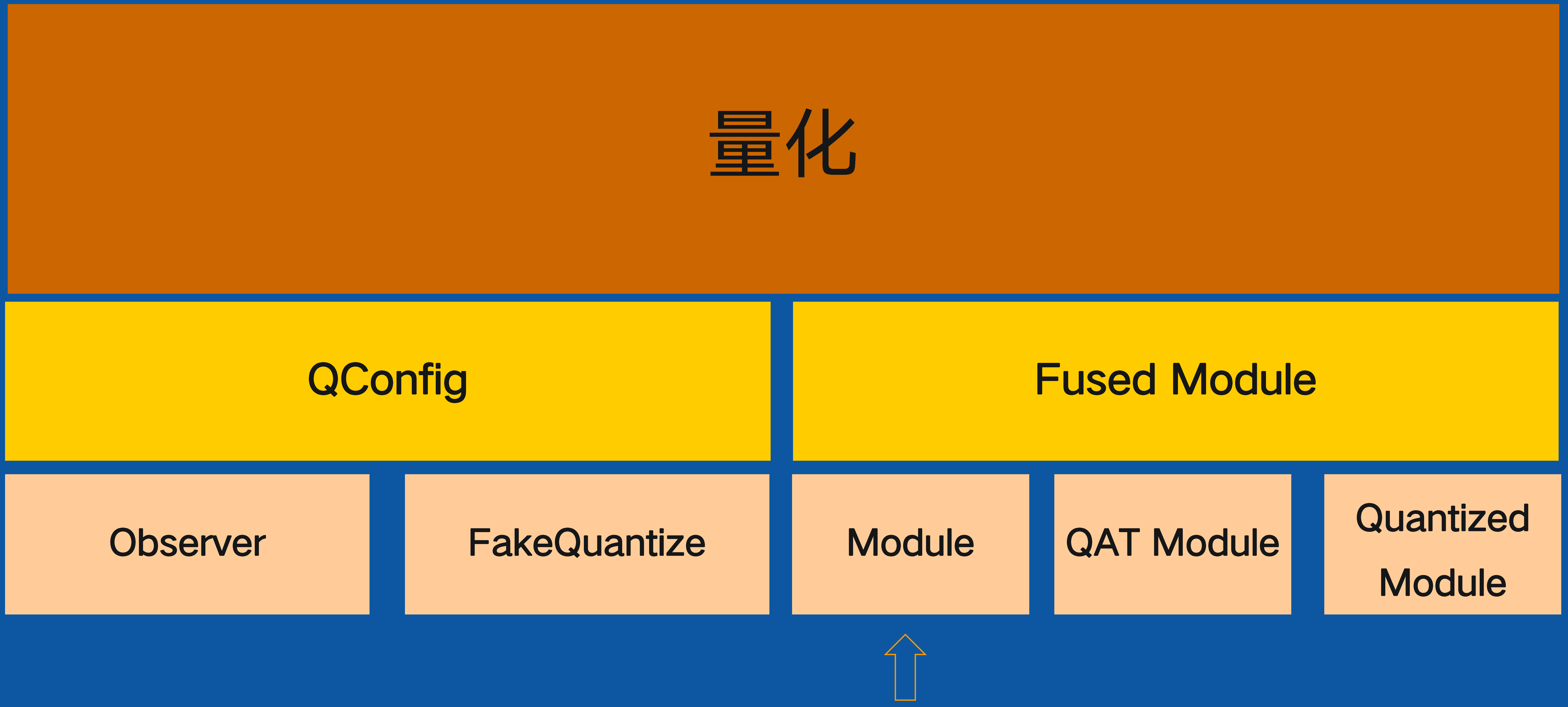
class megengine.quantization.observer.**Observer**

Bases: megengine.module.module.Module

A base class for Observer Module.

collect scale and zero_point of which dtype

量化方法介绍



megengine.module.activation

```
class megengine.module.activation.LeakyReLU(negative_slope=0.01)
```

megengine.module.batchnorm

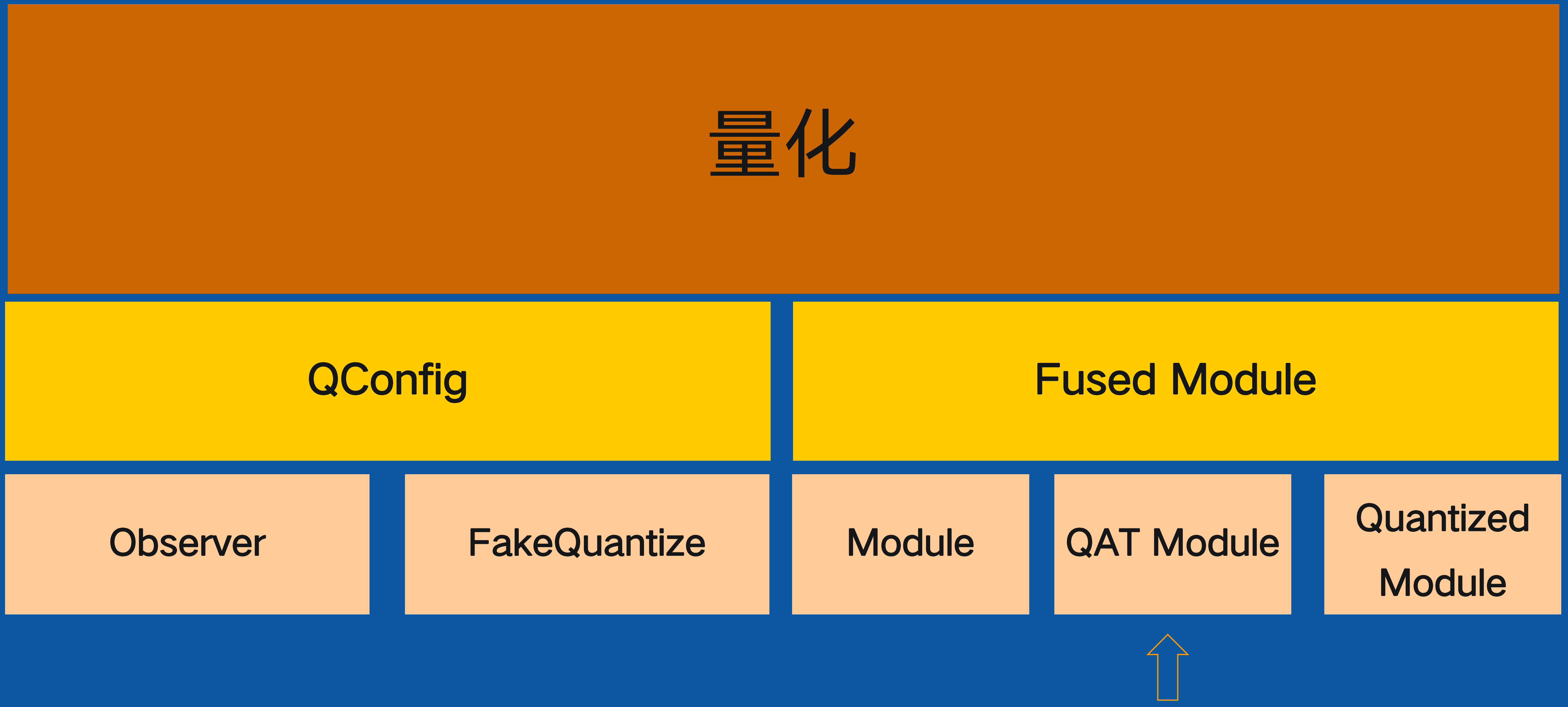
```
class megengine.module.batchnorm.BatchNorm1d(num_features, eps=1e-05, momentum=0.9, affine=True, track_running_stats=True)
```

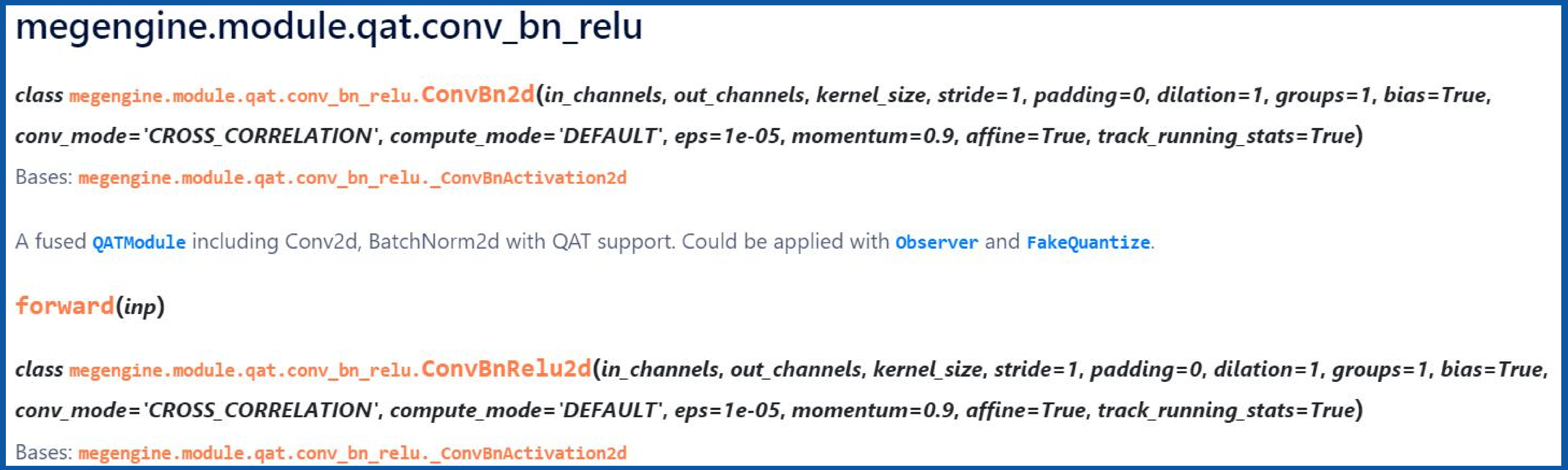
megengine.module.conv

```
class megengine.module.conv.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, conv_mode='CROSS_CORRELATION', compute_mode='DEFAULT')
```

量化方法介绍

MEGVII 旷视





megengine.module.qat.conv_bn_relu

class megengine.module.qat.conv_bn_relu.**ConvBn2d**(*in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, conv_mode='CROSS_CORRELATION', compute_mode='DEFAULT', eps=1e-05, momentum=0.9, affine=True, track_running_stats=True*)

Bases: megengine.module.qat.conv_bn_relu._ConvBnActivation2d

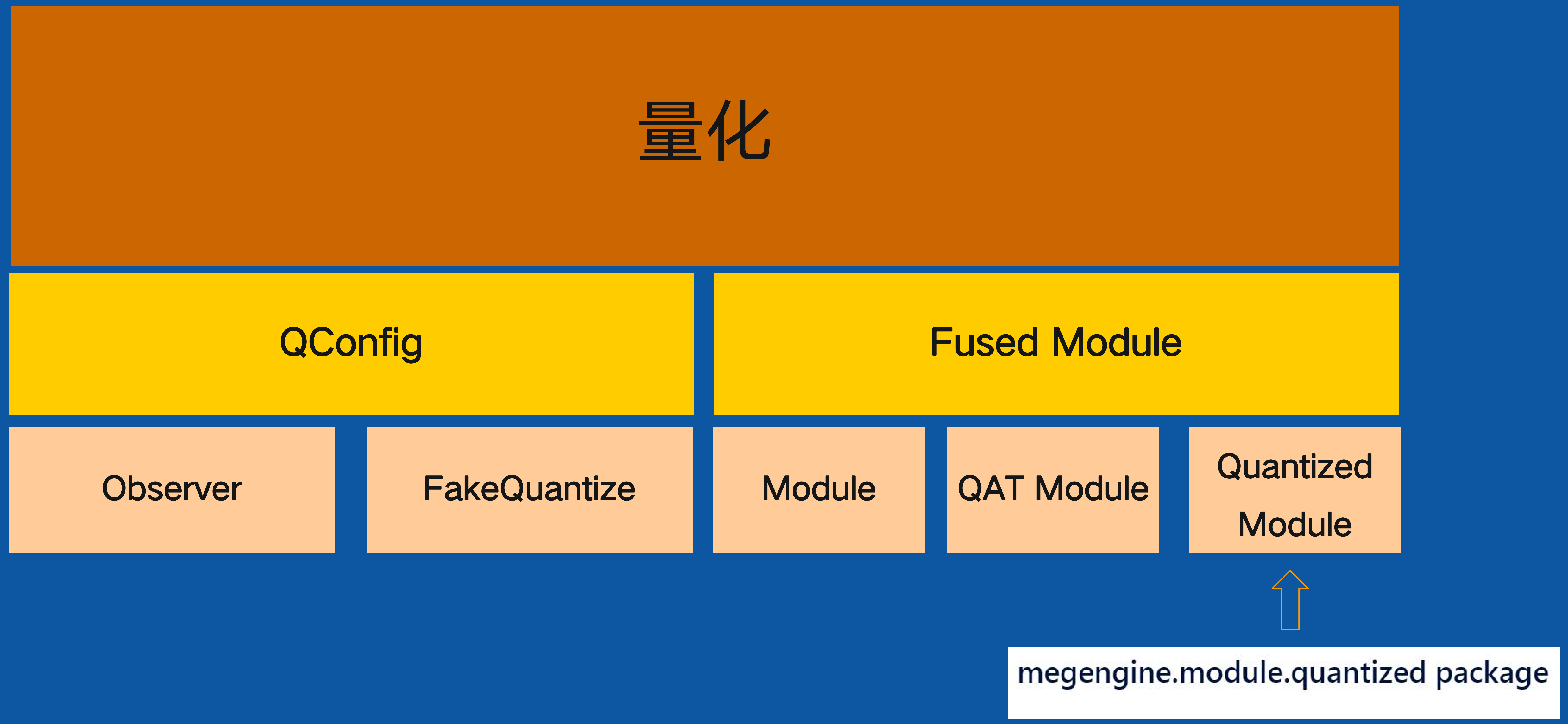
A fused QATModule including Conv2d, BatchNorm2d with QAT support. Could be applied with Observer and FakeQuantize.

forward(*inp*)

class megengine.module.qat.conv_bn_relu.**ConvBnRelu2d**(*in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, conv_mode='CROSS_CORRELATION', compute_mode='DEFAULT', eps=1e-05, momentum=0.9, affine=True, track_running_stats=True*)

Bases: megengine.module.qat.conv_bn_relu._ConvBnActivation2d

量化方法介绍



量化
QConfig
Fused Module
Observer
FakeQuantize
Module
QAT Module
Quantized Module
megengine.module.quantized package

MEGVII 旷视

1 概述

2 量化方法介绍

3 实例讲解

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
- 在正常模式下预训练模型，并在每轮迭代保存网络检查点；
- QAT 模式进行模型 Fine - tuning；
- 将模型转换为量化模型，并执行 Dump 用于后续模型部署。

```
(meg_engine) wangpeng@Amax:~/meg_test/Models/official/quantization$ ls -al
total 68
drwxrwxr-x 6 wangpeng wangpeng  4096 May 30 21:00 .
drwxrwxr-x 6 wangpeng wangpeng  4096 May 30 11:25 ..
-rwxrwxrwx 1 wangpeng wangpeng  1974 May 30 14:14 config.py
drwxrwxrwx 3 wangpeng wangpeng  4096 May 30 13:16 data
-rwxrwxrwx 1 wangpeng wangpeng 10887 May 30 16:13 finetune.py
-rwxrwxrwx 1 wangpeng wangpeng  3626 May 30 20:54 inference.py
drwxrwxrwx 3 wangpeng wangpeng  4096 May 30 20:35 models
drwxrwxrwx 2 wangpeng wangpeng  4096 May 30 14:17 __pycache__
-rwxrwxrwx 1 wangpeng wangpeng  2266 May 30 11:25 README.md
drwxrwxr-x 3 wangpeng wangpeng  4096 May 30 14:03 result
-rwxrwxrwx 1 wangpeng wangpeng  6219 May 30 11:25 test.py
-rwxrwxrwx 1 wangpeng wangpeng 10492 May 30 14:33 train.py
```

量化过程需要使用的脚本

天元开发者交流群

群号：1029741705

扫一扫二维码，加入群聊。

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
- 在正常模式下预训练模型，并在每轮迭代保存网络检查点；
- QAT 模式进行模型 Fine - tuning；
- 将模型转换为量化模型，并执行 Dump 用于后续模型部署。

```
wangpeng@Amax:~/meg_test/data$ tree -L 1
.
├── test
├── train
└── val
```

Flower数据集，共有104类待分类花卉

```
wangpeng@Amax:~/meg_test/data/train$ ls
 alpine_sea_holly    buttercup           cyclamen_            grape_hyacinth             mexican_petunia     prince_of_wales_feathers  thorn_apple
 anthurium           californian_poppy   daffodil             great_masterwort           monkshood           purple_coneflower         tiger_lily
 artichoke           camellia            daisy                hard-leaved_pocket_orchid  moon_orchid         red_ginger                toad_lily
 azalea              canna_lily          desert-rose          hibiscus                   morning_glory       rose                      tree_mallow
 balloon_flower      canterbury_bells    fire_lily            hippeastrum_               orange_dahlia       ruby-lipped_cattleya      tree_poppy
 barberton_daisy     cape_flower         foxglove             iris                       osteospermum        siam_tulip                trumpet_creeper
 bee_balm            carnation           frangipani           japanese_anemone           passion_flower      silverbush                wallflower
 bird_of_paradise    cautleya_spicata    fritillary           king_protea                peruvian_lily       snapdragon                watercress
 bishop_of_llandaff  clematis            garden_phlox         lenten_rose                petunia             spear_thistle             water_lily
 blackberry_lily    "colt's_foot"        gaura                lilac_hibiscus             pincushion_flower   spring_crocus             wild_geranium
 black-eyed_susan    columbine           gazania              lotus                      pink_primrose       stemless_gentian          wild_pansy
 blanket_flower      common_dandelion    geranium             love_in_the_mist           pink_quill          sunflower                 wild_rose
 bolero_deep_blue    common_tulip        giant_white_arum_lily magnolia                  pink-yellow_dahlia  sweet_pea                 windflower
 bougainvillea       corn_poppy          globe-flower         mallow                     poinsettia          sweet_william             yellow_iris
 bromelia            cosmos              globe_thistle        marigold                   primula             sword_lily
```

天元开发者交流群

群号：1029741705

扫一扫二维码，加入群聊。

实例讲解 - 以 ShufflenetV2 为例

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
  - 从使用下面命令 Github 上克隆 MegEngine/Models 仓库

    git clone https://github.com/MegEngine/Models.git

```
(meg_engine) wangpeng@Amax:~/meg_test$ git clone https://github.com/MegEngine/Models.git
Cloning into 'Models'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 276 (delta 1), reused 0 (delta 0), pack-reused 272
Receiving objects: 100% (276/276), 881.28 KiB | 208.00 KiB/s, done.
Resolving deltas: 100% (140/140), done.
```

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
  - 安装依赖包

    pip3 install --user -r requirements.txt

```
(meg_engine) wangpeng@Amax:~/meg_test/Models$ pip3 install --user -r requirements.txt
Collecting numpy (from -r requirements.txt (line 1))
  Using cached https://files.pythonhosted.org/packages/03/27/e35e7c6e6a52fab9fcc64fc2b2
Collecting opencv-python (from -r requirements.txt (line 2))
  Cache entry deserialization failed, entry ignored
  Downloading https://files.pythonhosted.org/packages/72/c2/e9cf54ae5b1102020ef895866a6
    100% |████████████████████████████████| 28.2MB 48kB/s
Collecting tqdm (from -r requirements.txt (line 3))
  Cache entry deserialization failed, entry ignored
  Cache entry deserialization failed, entry ignored
  Downloading https://files.pythonhosted.org/packages/c9/40/058b12e8ba10e35f89c9b1fdfc2
    100% |████████████████████████████████| 71kB 1.5MB/s
Collecting tabulate (from -r requirements.txt (line 4))
  Cache entry deserialization failed, entry ignored
  Cache entry deserialization failed, entry ignored
  Downloading https://files.pythonhosted.org/packages/c4/f4/770ae9385990f5a19a91431163d
Installing collected packages: numpy, opencv-python, tqdm, tabulate
Successfully installed numpy-1.18.4 opencv-python-4.2.0.34 tabulate-0.8.7 tqdm-4.46.0
```

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
  - 从当前目录进入到shufflenetv2源码所在目录，并查看

```
~/meg_test/Models$ cd official/vision/classification/shufflenet
~/meg_test/Models/official/vision/classification/shufflenet$ tree
```

```
.
├── inference.py
├── init.py
├── model.py
├── README.md
├── test.py
└── train.py

0 directories, 6 files
```

天元开发者交流群

群号：1029741705

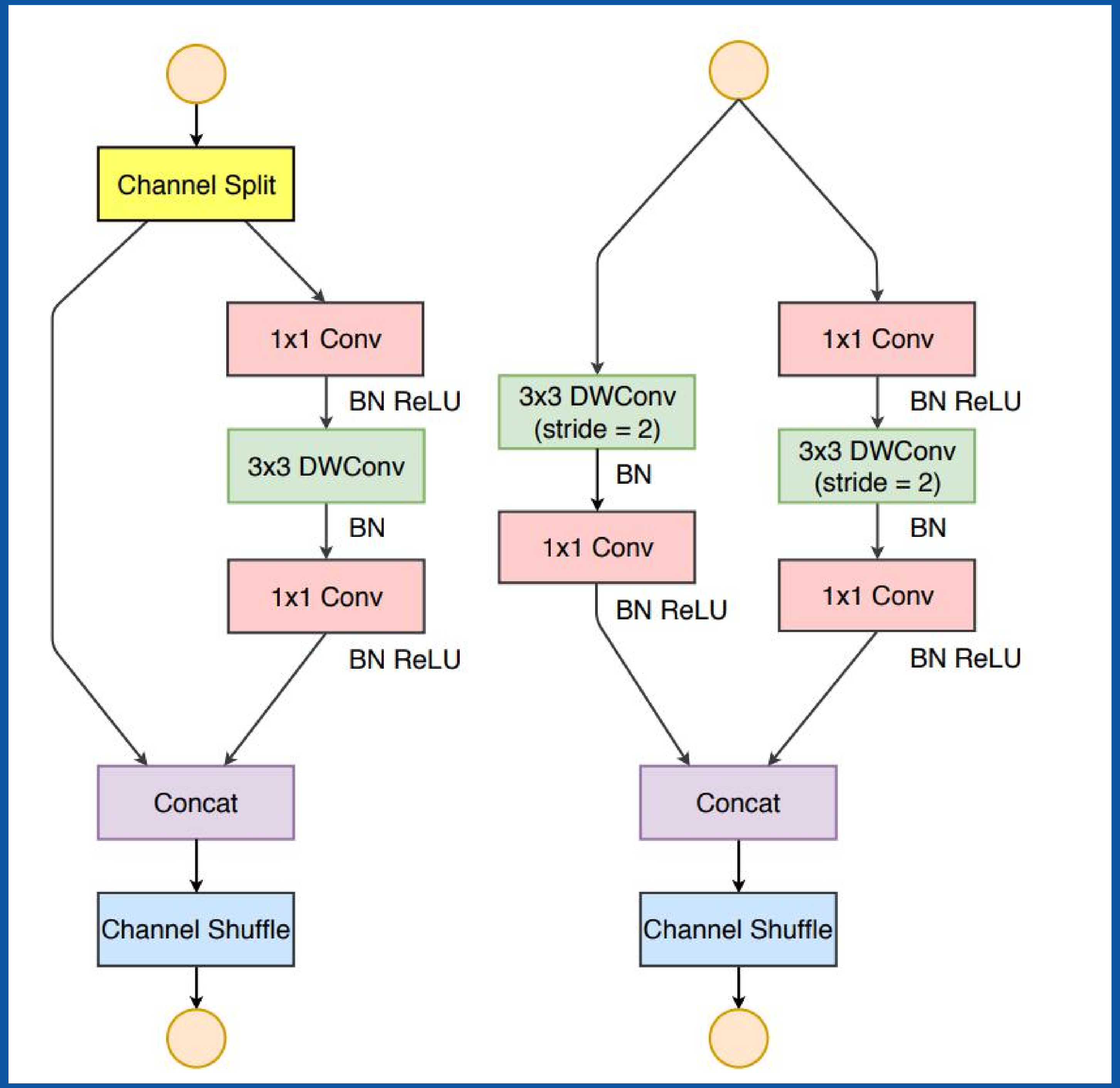
扫一扫二维码，加入群聊。

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
  - 打开 shufflenetV2 源码，将 Conv、Bn、Relu、Concat 等接口替换为QAT Module中的新接口



ShufflenetV2

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
  - 打开 shufflenetV2 源码，将 Conv、Bn、Relu、Concat 等接口替换为QAT Module中的新接口

megengine.module.qat.conv_bn_relu

class megengine.module.qat.conv_bn_relu.**ConvBn2d**(*in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=*
bias=True, conv_mode='CROSS_CORRELATION', compute_mode='DEFAULT', eps=1e-05, momentum=0.9, affine=True,
track_running_stats=True)

Bases: `megengine.module.qat.conv_bn_relu._ConvBnActivation2d`

A fused `QATModule` including Conv2d, BatchNorm2d with QAT support. Could be applied with `Observer` and `FakeQuantize`.

forward(*inp*)

class megengine.module.qat.conv_bn_relu.**ConvBnRelu2d**(*in_channels, out_channels, kernel_size, stride=1, padding=0, dila*
groups=1, bias=True, conv_mode='CROSS_CORRELATION', compute_mode='DEFAULT', eps=1e-05, momentum=0.9, affine
track_running_stats=True)

Bases: `megengine.module.qat.conv_bn_relu._ConvBnActivation2d`

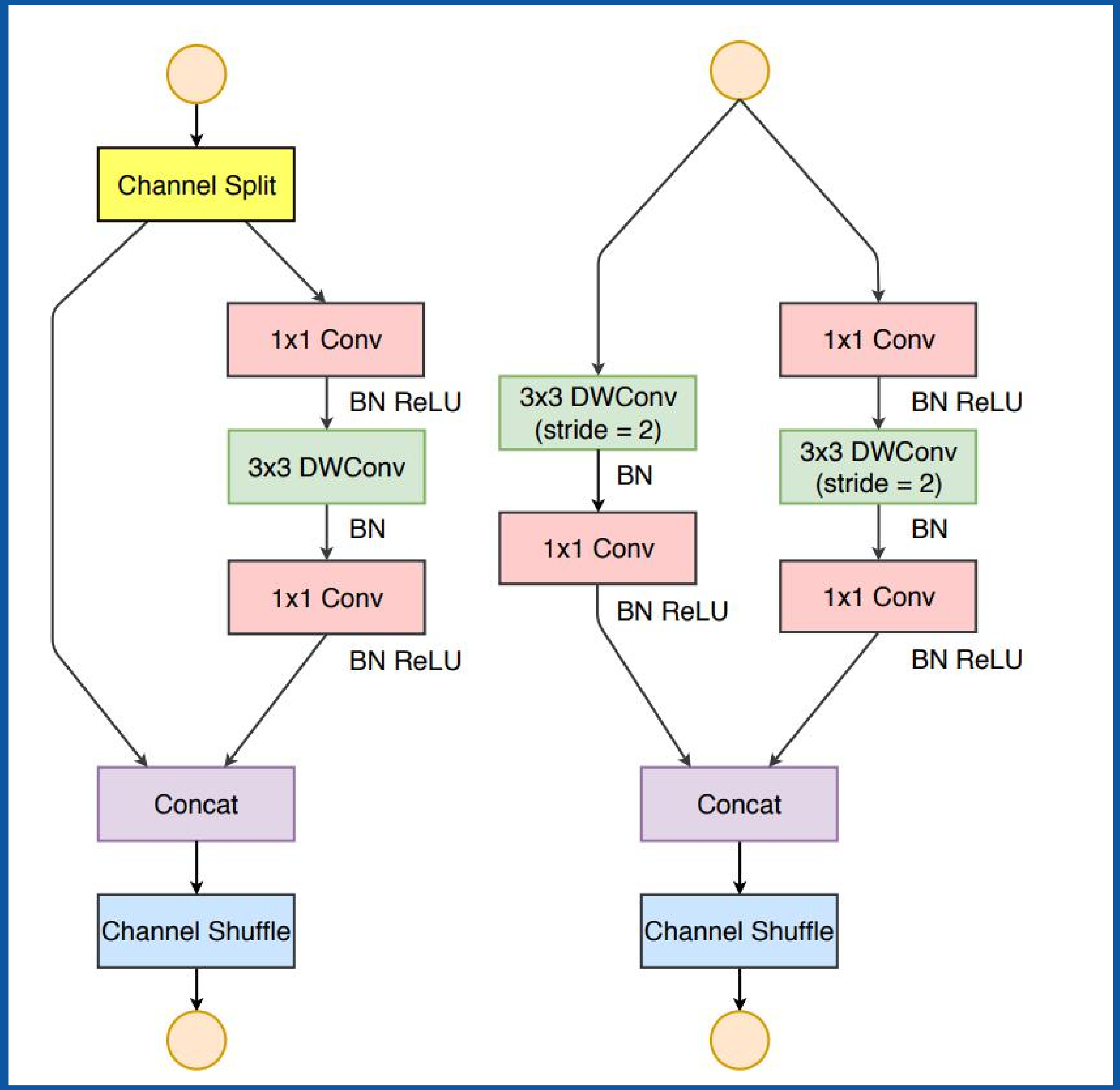
A fused `QATModule` including Conv2d, BatchNorm2d and relu with QAT support. Could be applied with `Observer` and `FakeQuantize`.

megengine.module.qat.concat

class megengine.module.qat.concat.**Concat**

Bases: `megengine.module.concat.Concat`, `megengine.module.qat.module.QATModule`

A `QATModule` to do functional concat with QAT support. Could be applied with `Observer` and `FakeQuantize`.



ShufflenetV2

实例讲解 - 以 ShufflenetV2 为例

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
  - 打开 shufflenetV2 源码，将 Conv、Bn、Relu、Concat 等接口替换为QAT Module中的新接口

```
def forward(self, x):
    x = self.first_conv(x)
    x = self.maxpool(x)
    x = self.features(x)
    x = self.conv_last(x)

    x = self.globalpool(x)
    if self.model_size == "2.0x":
        x = self.dropout(x)
    x = x.reshape(-1, self.stage_out_channels[-1])
    x = self.classifier(x)
    return x
```

实例讲解 - 以 ShufflenetV2 为例

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
  - 打开 shufflenetV2 源码，将 Conv、Bn、Relu、Concat 等接口替换为QAT Module中的新接口

```
def forward(self, x):
    x = self.first_conv(x)
    x = self.maxpool(x)
    x = self.features(x)
    x = self.conv_last(x)

    x = self.globalpool(x)
    if self.model_size == "2.0x":
        x = self.dropout(x)
    x = x.reshape(-1, self.stage_out_channels[-1])
    x = self.classifier(x)
    return x
```

```
    # building first layer
    input_channel = self.stage_out_channels[1]
    self.first_conv = M.Sequential(
        M.Conv2d(3, input_channel, 3, 2, 1, bias=False),
        M.BatchNorm2d(input_channel),
        M.ReLU(),
    )
```

实例讲解 - 以 ShufflenetV2 为例

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
  - 打开 shufflenetV2 源码，将 Conv、Bn、Relu、Concat 等接口替换为QAT Module中的新接口

```
def forward(self, x):
    x = self.first_conv(x)
    x = self.maxpool(x)
    x = self.features(x)
    x = self.conv_last(x)

    x = self.globalpool(x)
    if self.model_size == "2.0x":
        x = self.dropout(x)
    x = x.reshape(-1, self.stage_out_channels[-1])
    x = self.classifier(x)
    return x
```

```
self.features = []
for idxstage in range(len(self.stage_repeats)):
    numrepeat = self.stage_repeats[idxstage]
    output_channel = self.stage_out_channels[idxstage + 2]

    for i in range(numrepeat):
        if i == 0:
            self.features.append(
                ShuffleV2Block(
                    input_channel,
                    output_channel,
                    mid_channels=output_channel // 2,
                    ksize=3,
                    stride=2,
                )
            )
        else:
            self.features.append(
                ShuffleV2Block(
                    input_channel // 2,
                    output_channel,
                    mid_channels=output_channel // 2,
                    ksize=3,
                    stride=1,
                )
            )

        input_channel = output_channel

self.features = M.Sequential(*self.features)
```

实例讲解 - 以 ShufflenetV2 为例

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
  - 打开 shufflenetV2 源码，将 Conv、Bn、Relu 、Concat 等接口替换为QAT Module中的新接口

```python
def forward(self, x):
    x = self.first_conv(x)
    x = self.maxpool(x)
    x = self.features(x)
    x = self.conv_last(x)

    x = self.globalpool(x)
    if self.model_size == "2.0x":
        x = self.dropout(x)
    x = x.reshape(-1, self.stage_out_channels[-1])
    x = self.classifier(x)
    return x
```

```python
    self.conv_last = M.Sequential(
        M.Conv2d(input_channel, self.stage_out_channels[-1], 1, 1, 0, bias=False),
        M.BatchNorm2d(self.stage_out_channels[-1]),
        M.ReLU(),
    )
```

实例讲解 - 以 ShufflenetV2 为例

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
  - 打开 shufflenetV2 源码，将 Conv、Bn、Relu 、Concat 等接口替换为QAT Module中的新接口

```
def forward(self, old_x):
    if self.stride == 1:
        x_proj, x = self.channel_shuffle(old_x)
        return F.concat((x_proj, self.branch_main(x)), 1)
    elif self.stride == 2:
        x_proj = old_x
        x = old_x
        return F.concat((self.branch_proj(x_proj), self.branch_main(x)), 1)
    else:
        raise ValueError("use stride 1 or 2, current stride {}".format(self.stride))
```

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
  - 打开 shufflenetV2 源码，将 Conv、Bn、Relu 、Concat 等接口替换为QAT Module中的新接口

```diff
@@ -54,11 +70,9 @@ class ShuffleV2Block(M.Module):

         branch_main = [
             # pw
-            M.Conv2d(inp, mid_channels, 1, 1, 0, bias=False),
-            M.BatchNorm2d(mid_channels),
-            M.ReLU(),
+            ConvBnRelu2d(inp, mid_channels, 1, 1, 0, bias=False),
             # dw
-            M.Conv2d(
+            ConvBn2d(
                 mid_channels,
                 mid_channels,
                 ksize,
@@ -67,23 +81,17 @@ class ShuffleV2Block(M.Module):
                 groups=mid_channels,
                 bias=False,
             ),
-            M.BatchNorm2d(mid_channels),
             # pw-linear
-            M.Conv2d(mid_channels, outputs, 1, 1, 0, bias=False),
-            M.BatchNorm2d(outputs),
-            M.ReLU(),
+            ConvBn2d(mid_channels, outputs, 1, 1, 0, bias=False),
         ]
         self.branch_main = M.Sequential(*branch_main)

         if stride == 2:
             branch_proj = [
                 # dw
-                M.Conv2d(inp, inp, ksize, stride, pad, groups=inp, bias=False),
-                M.BatchNorm2d(inp),
+                ConvBn2d(inp, inp, ksize, stride, pad, groups=inp, bias=False),
                 # pw-linear
-                M.Conv2d(inp, inp, 1, 1, 0, bias=False),
-                M.BatchNorm2d(inp),
-                M.ReLU(),
+                ConvBnRelu2d(inp, inp, 1, 1, 0, bias=False),
             ]
             self.branch_proj = M.Sequential(*branch_proj)
         else:
```

修改前

修改后

天元开发者交流群

群号：1029741705

扫一扫二维码，加入群聊。

实例讲解 - 以 ShufflenetV2 为例

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
  - 打开 shufflenetV2 源码，将 Conv、Bn、Relu 、Concat 等接口替换为QAT Module中的新接口

```diff
@@ -129,9 +121,7 @@ class ShuffleNetV2(M.Module):
         # building first layer
         input_channel = self.stage_out_channels[1]
         self.first_conv = M.Sequential(
-            M.Conv2d(3, input_channel, 3, 2, 1, bias=False),
-            M.BatchNorm2d(input_channel),
-            M.ReLU(),
+            ConvBnRelu2d(3, input_channel, 3, 2, 1, bias=False),
         )

         self.maxpool = M.MaxPool2d(kernel_size=3, stride=2, padding=1)
@@ -168,9 +158,7 @@ class ShuffleNetV2(M.Module):
         self.features = M.Sequential(*self.features)

         self.conv_last = M.Sequential(
-            M.Conv2d(input_channel, self.stage_out_channels[-1], 1, 1, 0, bias=False),
-            M.BatchNorm2d(self.stage_out_channels[-1]),
-            M.ReLU(),
+            ConvBnRelu2d(input_channel, self.stage_out_channels[-1], 1, 1, 0, bias=False),
         )
         self.globalpool = M.AvgPool2d(7)
         if self.model_size == "2.0x":
```

修改前

修改后

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
  - 打开 shufflenetV2 源码，将 Conv、Bn、Relu 、Concat 等接口替换为QAT Module中的新接口

```
@@ -50,15 +66,16 @@ class ShuffleV2Block(M.Module):
         self.pad = pad
         self.inp = inp
+
+        self.concat=Concat()
+
         outputs = oup - inp

         branch_main = [
```

修改前

修改后

```
@@ -92,11 +103,11 @@ class ShuffleV2Block(M.Module):
     def forward(self, old_x):
         if self.stride == 1:
             x_proj, x = self.channel_shuffle(old_x)
-            return F.concat((x_proj, self.branch_main(x)), 1)
+            return self.concat((x_proj, self.branch_main(x)), 1)
         elif self.stride == 2:
             x_proj = old_x
             x = old_x
-            return F.concat((self.branch_proj(x_proj), self.branch_main(x)), 1)
+            return self.concat((self.branch_proj(x_proj), self.branch_main(x)), 1)
         else:
             raise ValueError("use stride 1 or 2, current stride {}".format(self.stride))
```

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
  - 加入数据转换接口

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
  - 加入数据转换接口

```
@@ -168,10 +177,11 @@ class ShuffleNetV2(M.Module):
         self.features = M.Sequential(*self.features)

         self.conv_last = M.Sequential(
-            M.Conv2d(input_channel, self.stage_out_channels[-1], 1, 1, 0, bias=False),
-            M.BatchNorm2d(self.stage_out_channels[-1]),
-            M.ReLU(),
+            ConvBnRelu2d(input_channel, self.stage_out_channels[-1], 1, 1, 0, bias=False),
         )
+
+        self.quant = QuantStub()
+        self.dequant = DequantStub()
```

```
     def forward(self, x):
+        x = self.quant(x)
         x = self.first_conv(x)
         x = self.maxpool(x)
         x = self.features(x)
         x = self.conv_last(x)

         x = self.globalpool(x)
+        x = self.dequant(x)
         if self.model_size == "2.0x":
             x = self.dropout(x)
         x = x.reshape(-1, self.stage_out_channels[-1])
```

天元开发者交流群

群号：1029741705

扫一扫二维码，加入群聊。

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
  - 加入数据转换接口

```diff
@@ -168,10 +177,11 @@ class ShuffleNetV2(M.Module):
         self.features = M.Sequential(*self.features)

         self.conv_last = M.Sequential(
-            M.Conv2d(input_channel, self.stage_out_channels[-1], 1, 1, 0, bias=False),
-            M.BatchNorm2d(self.stage_out_channels[-1]),
-            M.ReLU(),
+            ConvBnRelu2d(input_channel, self.stage_out_channels[-1], 1, 1, 0, bias=False),
         )
+
+        self.quant = QuantStub()
+        self.dequant = DequantStub()
```

```diff
     def forward(self, x):
+        x = self.quant(x)
         x = self.first_conv(x)
         x = self.maxpool(x)
         x = self.features(x)
         x = self.conv_last(x)

         x = self.globalpool(x)
+        x = self.dequant(x)
         if self.model_size == "2.0x":
             x = self.dropout(x)
         x = x.reshape(-1, self.stage_out_channels[-1])
```

```python
from .module import Module


class QuantStub(Module):
    r"""
    A helper :class:`~.Module` simply returning input. Could be replaced with
    version :class:`~.qat.QuantStub` using :func:`~.quantize.quantize_qat`.
    """

    def forward(self, inp):
        return inp


class DequantStub(Module):
    r"""
    A helper :class:`~.Module` simply returning input. Could be replaced with
    version :class:`~.qat.DequantStub` using :func:`~.quantize.quantize_qat`.
    """

    def forward(self, inp):
        return inp
```

天元开发者交流群

群号：1029741705

扫一扫二维码，加入群聊。

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
  - 加入数据转换接口

```diff
@@ -168,10 +177,11 @@ class ShuffleNetV2(M.Module):
         self.features = M.Sequential(*self.features)

         self.conv_last = M.Sequential(
-            M.Conv2d(input_channel, self.stage_out_channels[-1], 1, 1, 0, bias=False),
-            M.BatchNorm2d(self.stage_out_channels[-1]),
-            M.ReLU(),
+            ConvBnRelu2d(input_channel, self.stage_out_channels[-1], 1, 1, 0, bias=False),
         )
+
+        self.quant = QuantStub()
+        self.dequant = DequantStub()
```

```diff
     def forward(self, x):
+        x = self.quant(x)
         x = self.first_conv(x)
         x = self.maxpool(x)
         x = self.features(x)
         x = self.conv_last(x)

         x = self.globalpool(x)
+        x = self.dequant(x)
         if self.model_size == "2.0x":
             x = self.dropout(x)
         x = x.reshape(-1, self.stage_out_channels[-1])
```

```python
from .module import QATModule


class QuantStub(Float.QuantStub, QATModule):
    r"""
    A helper QATModule simply return input, but will quantize
    input after converted to :class:`~.QuantizedModule`.
    """

    def forward(self, inp):
        return self.apply_quant_activation(inp)
```

天元开发者交流群

群号：1029741705

扫一扫二维码，加入群聊。

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
  - 加入数据转换接口

```
@@ -168,10 +177,11 @@ class ShuffleNetV2(M.Module):
         self.features = M.Sequential(*self.features)

         self.conv_last = M.Sequential(
-            M.Conv2d(input_channel, self.stage_out_channels[-1], 1, 1, 0, bias=False),
-            M.BatchNorm2d(self.stage_out_channels[-1]),
-            M.ReLU(),
+            ConvBnRelu2d(input_channel, self.stage_out_channels[-1], 1, 1, 0, bias=False),
         )
+
+        self.quant = QuantStub()
+        self.dequant = DequantStub()
```

```
     def forward(self, x):
+        x = self.quant(x)
         x = self.first_conv(x)
         x = self.maxpool(x)
         x = self.features(x)
         x = self.conv_last(x)

         x = self.globalpool(x)
+        x = self.dequant(x)
         if self.model_size == "2.0x":
             x = self.dropout(x)
         x = x.reshape(-1, self.stage_out_channels[-1])
```

```
from .module import QATModule


class QuantStub(Float.
    r"""
    A helper QATModule
    input after conver
    """

    def forward(self,
        return self.ap
```

```
def forward(self, inp, q_dict):
    if self.enabled:
        if q_dict["mode"] == ObserverMode.SYMMERTIC:
            scale = q_dict["scale"]
            # Quant
            oup = Round()(inp / scale)
            # clip
            oup = F.minimum(F.maximum(oup, self.qmin), self.qmax)
            # DeQuant
            oup = (oup) * scale
            return oup
        else:
            scale = q_dict["scale"]
            zero_point = q_dict["zero_point"]
            # Quant
            oup = Round()(inp / scale) + zero_point
            # clip
            oup = F.minimum(F.maximum(oup, self.qmin), self.qmax)
            # DeQuant
            oup = (oup - zero_point) * scale
            return oup
    return inp
```

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
  - 加入数据转换接口

```
@@ -168,10 +177,11 @@ class ShuffleNetV2(M.Module):
         self.features = M.Sequential(*self.features)

         self.conv_last = M.Sequential(
-            M.Conv2d(input_channel, self.stage_out_channels[-1], 1, 1, 0, bias=False),
-            M.BatchNorm2d(self.stage_out_channels[-1]),
-            M.ReLU(),
+            ConvBnRelu2d(input_channel, self.stage_out_channels[-1], 1, 1, 0, bias=False),
         )
+
+        self.quant = QuantStub()
+        self.dequant = DequantStub()
```

```
     def forward(self, x):
+        x = self.quant(x)
         x = self.first_conv(x)
         x = self.maxpool(x)
         x = self.features(x)
         x = self.conv_last(x)

         x = self.globalpool(x)
+        x = self.dequant(x)
         if self.model_size == "2.0x":
             x = self.dropout(x)
         x = x.reshape(-1, self.stage_out_channels[-1])
```

```python
from .module import QuantizedModule


class QuantStub(QuantizedModule):
    r"""
    A helper quantize operation on input and inference only.
    """

    def __init__(self, dtype=None):
        super().__init__()
        self.output_dtype = dtype

    def forward(self, inp):
        return inp.astype(self.output_dtype)
```

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
- 在正常模式下预训练模型，并在每轮迭代保存网络检查点；

train.py 脚本默认情况下为加载ImageNet数据集的接口，使用Flower数据集，需要修改数据加载接口

修改前

修改后

```
@@ -149,11 +151,11 @@ def worker(rank, world_size, args):

     # Build train and valid datasets
     logger.info("preparing dataset..")
-    train_dataset = data.dataset.ImageNet(args.data, train=True)
-    train_sampler = data.Infinite(data.RandomSampler(
+    train_dataset = ImageFolder(root= os.path.join(args.data ,'train'))
+    train_sampler =  data.Infinite(data.RandomSampler(
         train_dataset, batch_size=cfg.BATCH_SIZE, drop_last=True
     ))
-    train_queue = data.DataLoader(
+    train_queue =  data.DataLoader(
         train_dataset,
         sampler=train_sampler,
         transform=T.Compose(
@@ -166,12 +168,13 @@ def worker(rank, world_size, args):
             ]
         ),
         num_workers=args.workers,
-    )
-    train_queue = iter(train_queue)
-    valid_dataset = data.dataset.ImageNet(args.data, train=False)
+     )
+    train_queue = iter(train_queue)
+    valid_dataset = ImageFolder(root= os.path.join(args.data ,'val')) #data.dataset.ImageNet(args.da
     valid_sampler = data.SequentialSampler(
         valid_dataset, batch_size=100, drop_last=False
     )
+
     valid_queue = data.DataLoader(
         valid_dataset,
         sampler=valid_sampler,
```

megengine.data.dataset.vision.folder

class `megengine.data.dataset.vision.folder.ImageFolder`(*root, check_valid_func=None, class_name=False*)

Bases: `megengine.data.dataset.vision.meta_vision.VisionDataset`

`__init__`(*root, check_valid_func=None, class_name=False*)

ImageFolder is a class for loading image data and labels from a organized folder.

the folder is expected to be organized as followed root/cls/xxx.img_ext

labels are indices of sorted classes in the root directory

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
- 在正常模式下预训练模型，并在每轮迭代保存网络检查点；

同时Flower数据集的花卉有104种，需要修改模型参数 num_classes = 104

```
@@ -242,5 +254,5 @@ def shufflenet_v2_x1_0(num_classes=1000):
 @hub.pretrained(
     "https://data.megengine.org.cn/models/weights/snetv2_x0_5_60750_c28db1a2.pkl"
 )
-def shufflenet_v2_x0_5(num_classes=1000):
+def shufflenet_v2_x0_5(num_classes=104):
     return ShuffleNetV2(num_classes=num_classes, model_size="0.5x")
```

修改前

修改后

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
- 在正常模式下预训练模型，并在每轮迭代保存网络检查点；

在目录~/meg_test/Models/official/quantization$ 执行以下命令开始训练

python3 train.py -a shufflenet_v2_x0_5 -d /home/wangpeng/meg_test/data --mode normal --save ./result/model/

其中

- a 表示模型名字

- d 表示数据集路径

-- mode 表示训练模式，normal表示常规训练

-- save 表示模型保存路径

注意：模型结构默认存放路径为：Models/official/quantization/model/ 并在 __init__.py 加载

```
.
├── __init__.py
├── mobilenet_v2.py
├── __pycache__
│   ├── __init__.cpython-37.pyc
│   ├── mobilenet_v2.cpython-37.pyc
│   ├── resnet.cpython-37.pyc
│   ├── shufflenet.cpython-37.pyc
│   └── shufflenet_v2.cpython-37.pyc
├── resnet.py
├── shufflenet.py
└── shufflenet_v2.py
```

```
from .resnet import *
from .shufflenet import *
from .mobilenet_v2 import *
from .shufflenet_v2 import *
```

天元开发者交流群

群号：1029741705

扫一扫二维码，加入群聊。

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
- 在正常模式下预训练模型，并在每轮迭代保存网络检查点；

在目录~/meg_test/Models/official/quantization$ 执行以下命令开始训练

python3 train.py -a shufflenet_v2_x0_5 -d /home/wangpeng/meg_test/data --mode normal --save ./result/model/

```
14:27:06 TRAIN e0 000000 0.500000 Loss 4.689 (4.689) Acc@1 4.004 (4.004) Acc@5 10.352 (10.352) Time 14.944 (14.944)
14:27:06 SAVING 000000
14:27:56 TRAIN e0 000050 0.498000 Loss 4.277 (4.639) Acc@1 5.566 (5.668) Acc@5 24.707 (21.717) Time 0.951 (1.006)
14:28:45 TRAIN e0 000100 0.496000 Loss 4.081 (4.179) Acc@1 9.375 (7.646) Acc@5 29.980 (27.736) Time 1.015 (0.961)
14:29:32 TRAIN e0 000150 0.494000 Loss 4.003 (4.033) Acc@1 9.668 (9.412) Acc@5 32.520 (32.047) Time 0.931 (0.952)
14:30:19 TRAIN e0 000200 0.492000 Loss 3.739 (3.924) Acc@1 14.453 (11.723) Acc@5 40.918 (35.531) Time 0.989 (0.946)
14:31:07 TRAIN e0 000250 0.490000 Loss 3.672 (3.716) Acc@1 16.992 (15.150) Acc@5 42.285 (42.217) Time 0.894 (0.950)
14:31:55 TRAIN e0 000300 0.488000 Loss 3.633 (3.605) Acc@1 14.453 (17.064) Acc@5 44.629 (46.393) Time 0.888 (0.953)
14:32:42 TRAIN e0 000350 0.486000 Loss 3.467 (3.529) Acc@1 20.898 (18.785) Acc@5 51.465 (48.973) Time 0.953 (0.950)
14:33:30 TRAIN e0 000400 0.484000 Loss 3.406 (3.452) Acc@1 21.973 (20.482) Acc@5 53.418 (51.441) Time 0.826 (0.951)
14:34:17 TRAIN e0 000450 0.482000 Loss 3.333 (3.392) Acc@1 23.535 (21.414) Acc@5 55.566 (53.635) Time 0.912 (0.950)
14:35:05 TRAIN e0 000500 0.480000 Loss 3.299 (3.319) Acc@1 24.121 (23.109) Acc@5 56.445 (56.170) Time 0.925 (0.961)
14:35:53 TRAIN e0 000550 0.478000 Loss 3.237 (3.260) Acc@1 24.902 (24.779) Acc@5 60.352 (57.846) Time 0.804 (0.951)
14:36:40 TRAIN e0 000600 0.476000 Loss 3.114 (3.214) Acc@1 29.199 (25.744) Acc@5 62.109 (59.242) Time 1.000 (0.948)
14:37:28 TRAIN e0 000650 0.474000 Loss 3.126 (3.155) Acc@1 28.320 (27.229) Acc@5 61.621 (61.479) Time 0.997 (0.947)
14:38:15 TRAIN e0 000700 0.472000 Loss 3.124 (3.111) Acc@1 27.441 (28.562) Acc@5 62.793 (62.756) Time 0.915 (0.952)
14:39:03 TRAIN e0 000750 0.470000 Loss 3.016 (3.038) Acc@1 31.152 (30.674) Acc@5 66.211 (64.893) Time 0.952 (0.954)
14:39:51 TRAIN e0 000800 0.468000 Loss 2.996 (2.992) Acc@1 30.469 (32.230) Acc@5 67.285 (66.252) Time 1.037 (0.959)
14:40:38 TRAIN e0 000850 0.466000 Loss 2.894 (2.931) Acc@1 35.059 (33.840) Acc@5 68.750 (67.865) Time 0.942 (0.946)
14:41:25 TRAIN e0 000900 0.464000 Loss 2.841 (2.874) Acc@1 37.598 (35.805) Acc@5 70.898 (69.822) Time 0.963 (0.946)
14:42:13 TRAIN e0 000950 0.462000 Loss 2.782 (2.813) Acc@1 39.648 (37.865) Acc@5 72.559 (71.039) Time 0.971 (0.953)
14:43:01 TRAIN e0 001000 0.460000 Loss 2.740 (2.761) Acc@1 40.723 (39.234) Acc@5 70.703 (72.174) Time 0.845 (0.952)
14:43:48 TRAIN e0 001050 0.458000 Loss 2.654 (2.710) Acc@1 43.359 (40.834) Acc@5 76.758 (73.773) Time 0.938 (0.945)
14:44:35 TRAIN e0 001100 0.456000 Loss 2.604 (2.668) Acc@1 44.531 (42.496) Acc@5 76.758 (74.631) Time 0.973 (0.950)
14:45:23 TRAIN e0 001150 0.454000 Loss 2.679 (2.616) Acc@1 42.676 (44.057) Acc@5 75.000 (76.109) Time 1.017 (0.950)
14:46:10 TRAIN e0 001200 0.452000 Loss 2.535 (2.554) Acc@1 47.266 (46.037) Acc@5 78.320 (77.283) Time 0.900 (0.947)
14:46:58 TRAIN e1 001250 0.450000 Loss 2.415 (2.514) Acc@1 50.879 (47.332) Acc@5 80.566 (77.914) Time 1.091 (0.950)
```

脚本会自动计算精度 top error 1 和 top error 5

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
- 在正常模式下预训练模型，并在每轮迭代保存网络检查点；

在目录~/meg_test/Models/official/quantization$ 执行以下命令开始训练

```
python3 train.py -a shufflenet_v2_x0_5 -d /home/wangpeng/meg_test/data --mode normal --save ./result/model/
```

```
15:46:30 TRAIN e4 005000 0.300000 Loss 1.318 (1.323) Acc@1 86.914 (86.623) Acc@5 97.461 (97.211) Time 1.230 (0.976)
15:47:18 TRAIN e4 005050 0.298000 Loss 1.343 (1.335) Acc@1 85.938 (86.320) Acc@5 97.559 (97.094) Time 0.910 (0.955)
15:48:06 TRAIN e4 005100 0.296000 Loss 1.296 (1.323) Acc@1 87.891 (86.602) Acc@5 97.070 (97.164) Time 0.947 (0.967)
15:48:54 TRAIN e4 005150 0.294000 Loss 1.283 (1.311) Acc@1 87.988 (86.969) Acc@5 97.949 (97.469) Time 0.934 (0.965)
15:49:43 TRAIN e4 005200 0.292000 Loss 1.301 (1.306) Acc@1 86.914 (87.229) Acc@5 97.656 (97.416) Time 1.031 (0.964)
15:50:30 TRAIN e4 005250 0.290000 Loss 1.304 (1.306) Acc@1 86.914 (87.182) Acc@5 97.656 (97.412) Time 0.951 (0.952)
15:51:18 TRAIN e4 005300 0.288000 Loss 1.304 (1.304) Acc@1 87.402 (87.293) Acc@5 97.070 (97.348) Time 1.013 (0.961)
15:52:06 TRAIN e4 005350 0.286000 Loss 1.282 (1.291) Acc@1 87.012 (87.875) Acc@5 97.559 (97.479) Time 0.988 (0.953)
15:52:53 TRAIN e4 005400 0.284000 Loss 1.272 (1.286) Acc@1 87.695 (87.859) Acc@5 98.242 (97.553) Time 0.923 (0.939)
15:53:40 TRAIN e4 005450 0.282000 Loss 1.266 (1.286) Acc@1 88.867 (88.016) Acc@5 97.949 (97.492) Time 0.976 (0.950)
15:54:28 TRAIN e4 005500 0.280000 Loss 1.262 (1.284) Acc@1 88.281 (87.920) Acc@5 98.340 (97.586) Time 0.822 (0.950)
15:55:15 TRAIN e4 005550 0.278000 Loss 1.251 (1.268) Acc@1 88.770 (88.539) Acc@5 97.656 (97.727) Time 0.979 (0.949)
15:56:02 TRAIN e4 005600 0.276000 Loss 1.253 (1.280) Acc@1 88.770 (88.041) Acc@5 97.559 (97.529) Time 0.965 (0.942)
15:56:50 TRAIN e4 005650 0.274000 Loss 1.272 (1.268) Acc@1 87.891 (88.570) Acc@5 98.047 (97.645) Time 0.971 (0.944)
15:57:37 TRAIN e4 005700 0.272000 Loss 1.267 (1.263) Acc@1 89.355 (88.643) Acc@5 97.949 (97.893) Time 1.016 (0.943)
15:58:24 TRAIN e4 005750 0.270000 Loss 1.226 (1.257) Acc@1 90.137 (88.842) Acc@5 97.949 (97.760) Time 0.963 (0.947)
15:59:11 TRAIN e4 005800 0.268000 Loss 1.278 (1.255) Acc@1 87.695 (89.006) Acc@5 97.754 (97.834) Time 0.936 (0.947)
```

epoch = 4 top error 1 和 top error 5

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
- 在正常模式下预训练模型，并在每轮迭代保存网络检查点；
- QAT 模式进行模型 Fine - tuning；

在Fine tuning时，也需要修改finetune.py中的数据加载接口

```diff
     logger.info("preparing dataset..")
-    train_dataset = data.dataset.ImageNet(args.data, train=True)
-    train_sampler = data.Infinite(data.RandomSampler(
+    train_dataset = ImageFolder(root= os.path.join(args.data ,'train'))
+    train_sampler =  data.Infinite(data.RandomSampler(
         train_dataset, batch_size=cfg.BATCH_SIZE, drop_last=True
     ))
-    train_queue = data.DataLoader(
+    train_queue =  data.DataLoader(
         train_dataset,
         sampler=train_sampler,
         transform=T.Compose(
@@ -174,12 +176,13 @@ def worker(rank, world_size, args):
             ]
         ),
         num_workers=args.workers,
-    )
+     )
     train_queue = iter(train_queue)
-    valid_dataset = data.dataset.ImageNet(args.data, train=False)
+    valid_dataset = ImageFolder(root= os.path.join(args.data ,'val')) #
     valid_sampler = data.SequentialSampler(
         valid_dataset, batch_size=100, drop_last=False
     )
+
     valid_queue = data.DataLoader(
         valid_dataset,
         sampler=valid_sampler,
@@ -194,6 +197,7 @@ def worker(rank, world_size, args):
         num_workers=args.workers,
     )

+
```

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
- 在正常模式下预训练模型，并在每轮迭代保存网络检查点；
- QAT 模式进行模型 Fine - tuning；

```
python3 finetune.py -a shufflenet_v2_x0_5 \
                    -d /home/wangpeng/meg_test/data \
                    --checkpoint ./result/model/shufflenet_v2_x0_5.normal/checkpoint.pkl \
                    --mode qat \
                    --save ./result/model/
```

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
- 在正常模式下预训练模型，并在每轮迭代保存网络检查点；
- QAT 模式进行模型 Fine - tuning；

```
python3 finetune.py -a shufflenet_v2_x0_5 \
                    -d /home/wangpeng/meg_test/data \
                    --checkpoint ./result/model/shufflenet_v2_x0_5.normal/checkpoint.pkl \
                    --mode qat \
                    --save ./result/model/
```

megengine.quantization.quantize.**quantize_qat**(*module, inplace=True, qconfig=<megengine.quantization.qconfig.QConfig object>*)

Recursively convert float `Module` to `QATModule` through `apply()` and set qconfig relatively.

Parameters:
- **module** (`Module`) – root module to do convert recursively.
- **inplace** – whether to convert submodules in-place.
- **qconfig** (`QConfig`) – an instance of `QConfig` to be set as submodules' qconfig. default is `ema_fakequant_qconfig`.

天元开发者交流群

群号：1029741705

扫一扫二维码，加入群聊。

实例讲解 - 以 ShufflenetV2 为例

MEGVII旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
- 在正常模式下预训练模型，并在每轮迭代保存网络检查点；
- QAT 模式进行模型 Fine - tuning；

```
python3 finetune.py -a shufflenet_v2_x0_5 \
                    -d /home/wangpeng/meg_test/data \
                    --checkpoint ./result/model/shufflenet_v2_x0_5.normal/checkpoint.pkl \
                    --mode qat \
                    --save ./result/model/
```

```
if args.mode != "normal":
    Q.quantize_qat(model, Q.ema_fakequant_qconfig)

if args.checkpoint:
    logger.info("Load pretrained weights from %s", args.checkpoint)
    ckpt = mge.load(args.checkpoint)
    ckpt = ckpt["state_dict"] if "state_dict" in ckpt else ckpt
    model.load_state_dict(ckpt, strict=False)
```

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
- 在正常模式下预训练模型，并在每轮迭代保存网络检查点；
- QAT 模式进行模型 Fine - tuning；
- 将模型转换为量化模型，并执行 Dump 用于后续模型部署。

```
python3 inference.py -a shufflenet_v2_x0_5 \
                     --checkpoint ./result/model/shufflenet_v2_x0_5.qat/checkpoint.pkl \
                     --mode quantized \
                     --dump
```

天元开发者交流群

群号：1029741705

扫一扫二维码，加入群聊。

实例讲解 - 以 ShufflenetV2 为例

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
- 在正常模式下预训练模型，并在每轮迭代保存网络检查点；
- QAT 模式进行模型 Fine - tuning；
- 将模型转换为量化模型，并执行 Dump 用于后续模型部署。

```
python3 inference.py -a shufflenet_v2_x0_5 \
                    --checkpoint ./result/model/shufflenet_v2_x0_5.qat/checkpoint.pkl \
                    --mode quantized \
                    --dump
```

`megengine.quantization.quantize.quantize(module, inplace=True)`

Recursively convert `QATModule` to `QuantizedModule` through `apply()`.

Parameters:
- **module** (`Module`) – root module to do convert recursively.
- **inplace** – whether to convert submodules in-place.

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
- 在正常模式下预训练模型，并在每轮迭代保存网络检查点；
- QAT 模式进行模型 Fine - tuning；
- 将模型转换为量化模型，并执行 Dump 用于后续模型部署。

```
python3 inference.py -a shufflenet_v2_x0_5 \
                     --checkpoint ./result/model/shufflenet_v2_x0_5.qat/checkpoint.pkl \
                     --mode quantized \
                     --dump
```

```
(meg_engine) wangpeng@Amax:~/meg_test/Models/official/quantization$ ls *.pkl -al
-rw-rw-r-- 1 wangpeng wangpeng 1846180 May 31 15:23 shufflenet_v2_x0_5.normal.pkl
-rw-rw-r-- 1 wangpeng wangpeng  788099 May 31 15:23 shufflenet_v2_x0_5.quantized.pkl
```

← 量化前

← 量化后

(1846180 – 788099) * 100 / 1846180= 57.3119

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
- 在正常模式下预训练模型，并在每轮迭代保存网络检查点；
- QAT 模式进行模型 Fine - tuning；
- 将模型转换为量化模型，并执行 Dump 用于后续模型部署。

```
python3 inference.py -a shufflenet_v2_x0_5 \
                     --checkpoint ./result/model/shufflenet_v2_x0_5.qat/checkpoint.pkl \
                     --mode quantized \
                     --dump
```

```
normal time =   37.6862 ms
quantized time =   16.1052 ms
```

← 量化前

← 量化后

（ 37.6862 － 16.1052 ）* 100 ／ 37.6862 ＝ 57. 2649

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

QAT模式量化步骤

- 基于Fused Module 修改网络结构；
- 在正常模式下预训练模型，并在每轮迭代保存网络检查点；
- QAT 模式进行模型 Fine - tuning；
- 将模型转换为量化模型，并执行 Dump 用于后续模型部署。

基于ImageNet数据集模型量化前后存储空间对比			
模型	量化前(MB)	量化后(MB)	降低百分比(%)
shufflenet_v1_x1.5	3.671	1.784	51.40
resnet18	19.673	9.258	52.94
mobilenet_v2	14.196	7.403	47.85

基于ImageNet数据集模型量化前后性能对比			
模型	量化前(ms)	量化后(ms)	提高百分比(%)
shufflenet_v1_x1.5	57.9	39.5	31.78
resnet18	95.4	61.5	35.53
mobilenet_v2	76.5	57.3	25.10

实例讲解 - 以 ShufflenetV2 为例

MEGVII 旷视

参考链接

- MegEngine/Models 地址： https://github.com/MegEngine/Models.git
- Flower数据集地址：https://www.kaggle.com/c/flower-classification-with-tpus/data
- 量化脚本文件地址：https://github.com/MegEngine/Models/tree/master/official/quantization
- Shufflenet_v2源码地址： https://github.com/MegEngine/Models/tree/master/official/vision/classification/shufflenet

作业

MEGVII 旷视

- 了解MegEngine框架下量化原理
- 利用MegEngine框架修改Shufflenet_v2模型为可量化模型
- 使用train.py、finetune.py 和inference.py 验证量化模型是否修改成功。

提交：

- 将修改后的模型py文件, 量化后的模型size加上以下信息发送到 mgesupport@megvii.com

邮件标题：天元入门第六次课程作业
姓名：
学校（公司）：
电话：
邮寄地址：

天元开发者交流群
群号：1029741705
扫一扫二维码，加入群聊。

- 本课程供有6节课程，课程相关资料（课件、视频、作业）可以从MegEngine的Github中获取

- 6次作业全部完成的同学，我们将为你发放【天元宇宙系列】大礼包！
- 考虑到9月份是返校季，我们作业回收截止日期延长到**9月15号**。
- 作业提交邮箱：mgesupport@megvii.com

MEGVII 旷视

欢迎加入“天元开发者交流群”

天元开发者交流群

群号：1029741705

扫一扫二维码，加入群聊。

行正则致远

AI向善，行胜于言。

MEGVII 旷视