1 Introduction to Quantization

1.1 Quantization Explanation

1.1.1 Quantization Definition

모델 양자화는 딥러닝 모델의 부동소수점 매개변수 및 연산을 고정소수점 표현으로 변환하는 과정을 의미합니다. 예를 들어 FLOAT32를 INT8로 변환하는 것이 이에 해당됩니다. 양자화는 메모리 사용량을 줄이고 모델 압축 및 추론 가속화를 달성할 수 있지만, 정확도 측면에서 일정 수준의 손실을 초래할 수 있습니다.

1.1.2 Quantization Calculation Principle

선형 비대칭 양자화를 예로 들면, 부동소수점 수를 부호 있는 고정소수점 수로 양자화 하는 계산 원리는 다음과 같습니다:

$$x_{int} = clamp(\lfloor \frac{x}{s} \rceil + z; -2^{b-1}, 2^{b-1} - 1) \tag{6-1}$$

x는 부동소수점 수, x_{int} 는 양자화 고정소수점 수, $\lfloor \cdot \rceil$ 는 반올림 연산을 나타내며, s 는 양자화 스케일 인자, z는 양자화 제로 포인트, b는 양자화 비트 너비(예: INT8 데이터 유형의 경우 8)입니다. Clamp는 절단 연산으로, 다음과 같이 정의됩니다.

$$clamp(x; a, c) = \begin{cases} a, & x < a, \\ x, & a \le x \le c, \\ c, & x > c, \end{cases} \tag{6-2}$$

고정소수점 수를 부동소수점 수로 변환하는 과정은 '탈양자화'라고 하며, 다음과 같이 정의됩니다:

$$x \approx \widehat{x} = s\left(x_{int} - z\right) \tag{6-3}$$

양자화 범위 $(q_{\min})$와 클리핑 범위 $(c_{\min})$가 주어졌을 때, 양자화 매개변수s 와 z의 계산 공식은 다음과 같습니다.

$$s = \frac{q_{\max} - q_{\min}}{c_{\max} - c_{\min}}$$
$$z = c_{\max} - \left\lfloor \frac{q_{\max}}{s} \right\rceil \text{ 或 } z = c_{\min} - \lfloor \frac{q_{\min}}{s} \rceil \tag{6-5}$$

정량화 데이터 유형에 따라 절단 범위가 결정됩니다. 예를 들어, INT8의 절단 범위는 (-128, 127)입니다. 양자화 범위는 제6.1.6절 '양자화 알고리즘'에 설명된 다양한 양자화 알고리즘에 따라

결정됩니다. 자세한 내용은 해당 절을 참조하십시오.

1.1.3 Quantization Error

양자화는 모델에 일정 수준의 정확도 손실을 초래합니다. 방정식 (6-1)에 따르면, 양자화 오차는 반올림 오차와 절삭 오차에서 비롯되며, 즉 $\lfloor \cdot \rceil$ 및 $clamp$ 연산에서 발생합니다. 반올림 연산은 반올림 오차를 발생시키며, 이 오차는 $\left(-\frac{1}{2}s, \frac{1}{2}s\right)$ 의 범위를 가집니다. 부동소수점 수 x가 너무 크고 스케일 인자가 너무 작을 경우, 양자화된 고정소수점 수가 절삭 범위를 초과하여 절삭 오차가 발생할 수 있습니다. 이론적으로, 스케일 팩터를 증가시키면 절삭 오류를 줄일 수 있지만 라운딩 오류를 증가시킬 수 있습니다. 따라서 두 오류를 균형 있게 조정하기 위해 적절한 스케일 팩터와 제로 포인트를 설계하여 양자화 오류를 최소화해야 합니다.

1.1.4 Linear Symmetric Quantization and Linear Asymmetric Quantization

선형 양자화에서 고정소수점 수 사이의 간격은 균일합니다. 예를 들어, INT8 선형 양자화는 양자화 범위를 256개의 수로 균등하게 나눕니다. 선형 대칭 양자화에서는 양자화 데이터 유형에 따라 0점이 결정되며, 양자화된 고정소수점 수 범위의 대칭 중심점에 위치합니다. 예를 들어, INT8의 0점은 0입니다. 선형 비대칭 양자화에서는 0점이 방정식 (6-5)에 따라 계산되며, 일반적으로 양자화된 고정소수점 수 범위 내의 대칭 중심점에 위치하지 않습니다. 대칭 양자화는 비대칭 양자화의 단순화된 버전입니다. 이론적으로 비대칭 양자화는 비균일한 데이터 분포를 더 잘 처리할 수 있으므로, 실제 적용에서는 일반적으로 비대칭 양자화가 사용됩니다.

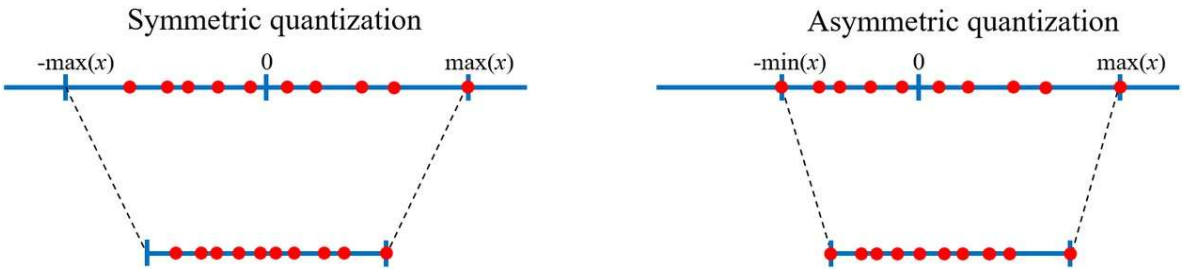

Figure 6-1 Linear Symmetric Quantization and Linear Asymmetric Quantization

1.1.5 Per-Layer Quantization and Per-Channel Quantization

레이어별 양자화는 네트워크 레이어의 모든 채널을 전체적으로 양자화하며, 모든 채널이 동일한 양자화 매개변수를 공유합니다. 채널별 양자화는 네트워크 레이어의 각 채널을 독립적으로 양자화하며, 각 채널은 자체 양자화 매개변수를 갖습니다. 채널별 양자화는 각 채널의 정보를 더 잘 보존하며, 서로 다른 채널 간의 차이에 적응하고, 더 우수한 양자화 결과를 제공합니다.

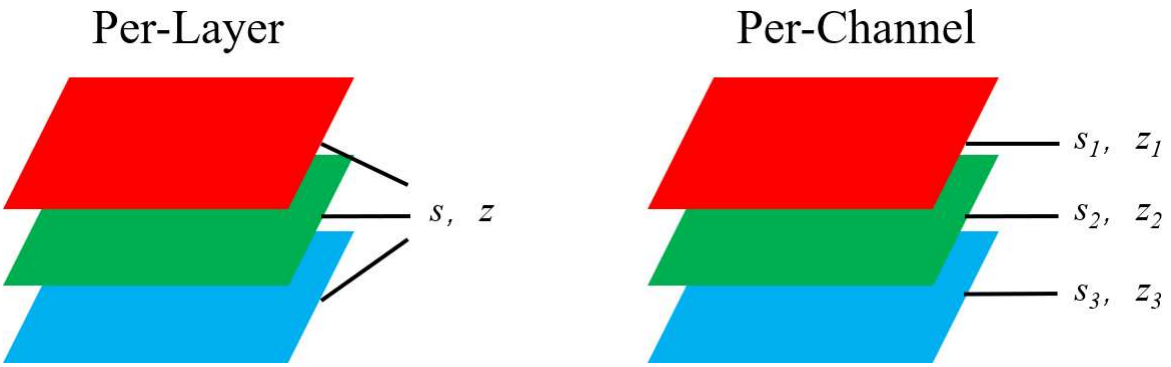

Figure 6-2 Per-Layer Quantization and Per-Channel Quantization

Note: RKNN-Toolkit2에서 채널별 양자화는 가중치에만 적용되며, 활성화 값과 중간 값은 여전히 레이어별 양자화를 사용하여 양자화됩니다.

1.1.6 Quantization Algorithms

양자화 척도 인자 및 제로 포인트는 양자화 오차에 영향을 미치는 중요한 매개변수이며, 양자화 범위 해결 방법은 양자화 매개변수를 결정하는 데 결정적인 역할을 합니다. 이 섹션에서는 양자화 범위를 해결하기 위한 세 가지 알고리즘을 소개합니다.

정상 양자화 알고리즘은 부동소수점 숫자의 최대값과 최소값을 직접 계산하여 양자화 범위의 최대값과 최소값을 결정합니다. 제6.1.2절의 양자화 계산 원리에서 알 수 있듯이, 정상 양자화 알고리즘은 절삭 오류를 도입하지 않지만 아웃라이어에 민감합니다. 이는 큰 아웃라이어가 심각한 반올림 오류를 유발할 수 있기 때문입니다.

$$q_{\min} = \min \mathbf{V} \tag{6-6}$$

$$q_{\max} = \max \mathbf{V} \tag{6-7}$$

V는 부동소수점 수 텐서입니다.

KL-Divergence 양자화 알고리즘은 부동소수점 수와 고정소수점 수의 분포를 계산합니다. 이 알고리즘은 다양한 임계 값을 조정하여 분포를 업데이트하며, 두 분포 간의 유사성을 최대화하기 위해 KL 분산을 최소화함으로써 양자화 범위의 최대값과 최소값을 결정합니다. KL-Divergence 양자화 알고리즘은 부동소수점 수와 고정소수점 수 간의 분포 차이를 최소화하여 비 균일한 데이터 분포에 더 잘 적응하고 소수의 아웃라이어의 영향을 완화합니다.

$$\arg\min_{q_{\min}, q_{\max}} H(\Psi(\mathbf{V}),\ \Psi(\mathbf{V}_{\text{int}})) \tag{6-8}$$

$H(\cdot)$ 는 KL 분산 계산 공식이며, $\Psi(\cdot)$ 는 해당 데이터를 이산 분포로 이산화하는 분포 함수이며, V_{int} 는 양자화된 고정 소수점 수 텐서입니다.

MMSE(최소 평균 제곱 오차) 양자화 알고리즘은 부동 소수점 수와 양자화 및 양자화 해제된 부동 소수점 수 사이의 평균 제곱 오차 손실을 최소화함으로써 양자화 범위의 최대값과 최소값을 결정합니다. 이 알고리즘은 큰 아웃라이어로 인해 발생하는 양자화 정확도 손실을 어느 정도 완화합니다. MMSE 양자화 알고리즘은 대략적인 해를 찾기 위해 무차별 대입 반복 검색을 통해

구현되므로 속도가 느리고 메모리 오버헤드가 높습니다. 그러나 일반적으로 일반 양자화 알고리즘에 비해 더 높은 양자화 정확도를 달성합니다.

$$\arg\min_{q_{\min},q_{\max}} \left\| \mathbf{V} - \widehat{\mathbf{V}}(q_{\min}, q_{\max}) \right\|_F^2 \tag{6-9}$$

여기서 v는 양자화 및 탈양자화된 형태를 나타내며, $||\ ||_F$ 은 Frobenius 노름을 의미합니다.

1.2 Quantization Configuration

1.2.1 Quantization Data Types

RKNN-Toolkit2는 INT8를 양자화 데이터 유형으로 지원합니다.

1.2.2 Quantization Algorithm Recommendations

정규 양자화 알고리즘은 실행 속도가 빠르고 일반적인 시나리오에 적합합니다.

KL-Divergence 양자화 알고리즘은 정규화 알고리즘보다 약간 더 느리게 실행됩니다. 이 알고리즘은 비균일 분포를 가진 모델의 양자화 정확도를 향상시키며, 소수의 아웃라이어 값으로 인한 양자화 정확도 손실을 완화할 수 있습니다.

MMSE 양자화 알고리즘은 더 느리게 실행되며 더 많은 메모리를 소비합니다. KL-Divergence 알고리즘과 비교할 때, 이상치 값으로 인한 양자화 정확도 손실을 더 잘 완화합니다. 양자화 친화적인 모델의 경우, 사용자는 양자화 정확도를 개선하기 위해 MMSE 양자화 알고리즘을 시도해 볼 수 있습니다. 대부분의 시나리오에서 MMSE 알고리즘은 Normal 및 KL-Divergence 알고리즘보다 더 높은 양자화 정확도를 제공합니다.

정규 양자화 알고리즘은 기본으로 사용됩니다. 양자화 정확도 문제가 발생할 경우 사용자는 KL-Divergence 및 MMSE 양자화 알고리즘을 시도해 볼 수 있습니다.

1.2.3 Quantization Calibration Set Recommendations

양자화 캘리브레이션 세트는 활성화 값의 양자화 범위를 계산하는 데 사용됩니다. 양자화 캘리브레이션 세트를 선택할 때는 모델의 실제 적용 시나리오에서 발생하는 다양한 데이터 분포를 포함해야 합니다. 예를 들어, 분류 모델의 경우 양자화 보정 세트에는 실제 적용 시나리오에서 다양한 카테고리의 이미지가 포함되어야 합니다. 일반적으로 양자화 보정 세트는 20~200개의 이미지로 구성하는 것이 권장되며, 이는 양자화 알고리즘의 실행 시간에 따라 조정될 수 있습니다. 양자화 보정 세트의 이미지 수를 늘리면 양자화 알고리즘의 실행 시간이 증가할 수 있지만, 반드시 양자화 정확도가 향상되는 것은 아니라는 점을 유의해야 합니다.

1.2.4 Quantization Configuration Method

RKNN-Toolkit2의 양자화 구성 방법은 rknn.config() 및 rknn.build() 인터페이스에 구현되어 있습니다. 양자화 방법 구성은 rknn.config() 인터페이스에 구현되어 있으며, 양자화 스위치 및 교정 세트 경로

선택은 rknn.build() 인터페이스에 구현되어 있습니다.
rknn.config() 인터페이스에는 다음과 같은 관련 양자화 구성 옵션이 포함됩니다:

quantized_dtype: 양자화 데이터 유형을 선택합니다. 현재는 선형 비대칭 INT8 양자화만 지원되며, 기본값은 'asymmetric_quantized-8'입니다.
quantized_algorithm: 양자화 알고리즘을 선택합니다. Normal, KL-Divergence, MMSE 양자화 알고리즘이 포함됩니다. 선택 가능한 값은 'normal', 'kl_divergence', 'mmse'이며, 기본값은 'normal'입니다.
quantized_method: 레이어별 또는 채널별 양자화를 선택합니다. 선택 가능한 값은 'layer'와 'channel'이며, 기본값은 'channel'입니다.
rknn.build() 인터페이스에는 다음과 같은 관련 양자화 구성 옵션이 포함됩니다: do_quantization: 양자화를 활성화할지 여부를 지정합니다. 기본값은 'False'입니다.
dataset: 양자화 교정 세트의 경로를 지정합니다. 기본값은 'None'입니다.
현재 텍스트 파일 형식이 지원됩니다. 사용자는 교정용 이미지(jpg 또는 png 형식) 또는 npy 파일의 경로를 .txt 파일에 입력할 수 있습니다. 텍스트 파일의 각 행은 경로를 나타냅니다. 예를 들어:

```
a.jpg
b.jpg
```

여러 개의 입력 항목이 있는 경우, 각 입력 항목에 해당하는 파일은 공백으로 구분됩니다. 예를 들어:

```
a0.jpg a1.jpg
b0.jpg b1.jpg
```

1.3 Hybrid Quantization

하이브리드 양자화는 모델의 각 층에 서로 다른 양자화 데이터 유형을 적용합니다. 양자화에 적합하지 않은 층에는 양자화 정확도 손실을 완화하기 위해 더 높은 정밀도의 데이터 유형을 사용합니다.
그러나 하이브리드 정밀도 양자화는 추가적인 오버헤드를 증가시키며, 사용자가 각 층에 대한 양자화 데이터 유형을 결정해야 합니다.

1.3.1 Usage of Hybrid Quantization

성능과 정확도 사이의 균형을 더 잘 맞추기 위해 RKNN-Toolkit2는 하이브리드 양자화 기능을 제공합니다. 사용자는 정확도 분석의 결과에 따라 각 레이어를 양자화할지 여부를 수동으로 지정할 수 있습니다.
현재 하이브리드 양자화 기능은 다음과 같은 사용 사례를 지원합니다:

지정된 양자화 레이어를 비양자화 레이어로 변경하고 FLOAT16을 사용하여 계산합니다. (참고: NPU에서 비양자화 계산은 성능이 낮아 추론 속도가 감소됩니다.)
각 레이어의 양자화 매개변수도 수정할 수 있습니다. (양자화 매개변수를 수정하는 것은 권장되지 않습니다.)

1.3.2 Process of Hybrid Quantization Usage

하이브리드 양자화 기능을 사용할 때, 다음 네 가지 특정 단계가 포함됩니다.

1. 원본 모델을 로드하고 양자화 구성 파일, 임시 모델 파일 및 데이터 파일을 생성합니다.

   구체적인 인터페이스 호출 과정은 다음과 같습니다:

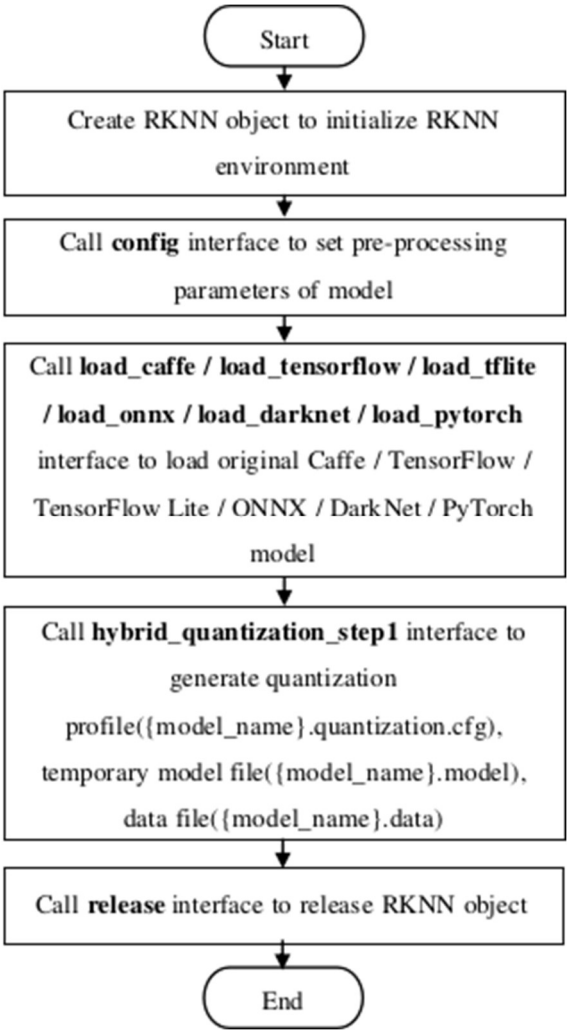

Figure 6-3 Hybrid quantization step 1

2. 첫 번째 단계에서 생성된 양자화 구성 파일을 수정합니다.

첫 번째 단계에서 hybrid_quantization_step1 인터페이스를 호출한 후, 현재 디렉토리에 {모델 이름}.quantization.cfg라는 이름의 구성 파일이 생성됩니다. 구성 파일의 형식은 다음과 같습니다:

```
custom_quantize_layers:
    Conv_ 350:0: float16
    Conv_ 358:0: float16
    ....
quantize_parameters:
    ....
    FeatureExtractor/MobilenetV2/Conv/Relu6:0:
        qtype: asymmetric_quantized
        qmethod: layer
        dtype: int8
        min:
        - 0.0
        max:
        - 6.0
        scale:
        - 0.023529411764705882
        zero_point:
        - -128
    ....
```

"custom_quantize_layers" 섹션에서 사용자는 텐서 이름을 지정하고 그 다음에 양자화 유형을 지정하여 사용자 정의 양자화 레이어를 추가할 수 있습니다. 이로 인해 해당 레이어의 연산 유형이 지정된 유형으로 변경됩니다. 현재 사용할 수 있는 양자화 데이터 유형은 "float16"입니다.

"quantize_parameters" 섹션에는 모델 내 각 텐서에 대한 양자화 매개변수가 포함되어 있습니다. 각 텐서의 양자화 매개변수는 "tensor_name: quantization_properties_and_parameters" 형식으로 표시됩니다. "min"과 "max"는 양자화 범위의 최소 및 최대 값을 나타냅니다. 사용자는 정밀도 분석의 출력 결과를 참조하거나 Netron을 사용하여 {model_name}.model이라는 임시 모델 파일을 열어 해당 출력 텐서 이름을 확인할 수 있습니다.

3. RKNN 모델을 생성합니다. 구체적인 인터페이스 호출 과정은 다음과 같습니다:

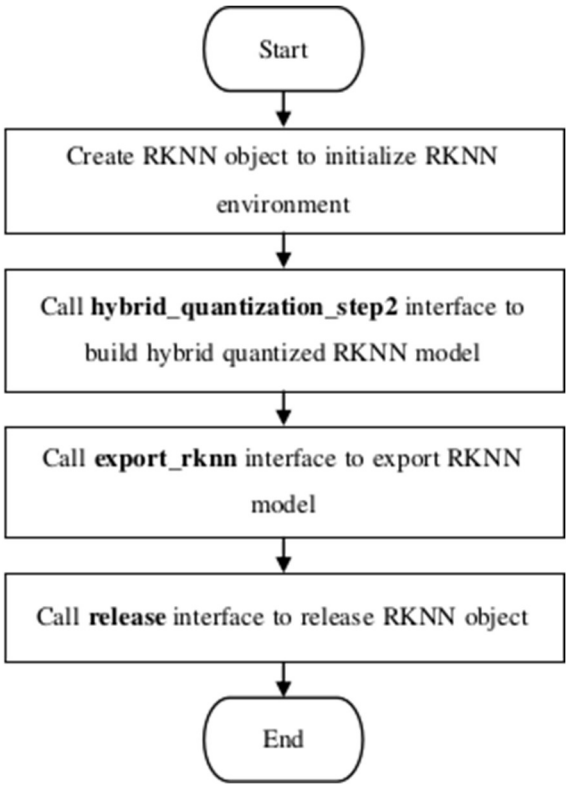

Figure 6-4 Hybrid quantization step 3

4. 제3단계에서 생성된 RKNN 모델을 사용하여 추론을 수행합니다.

참고: RKNN-Toolkit2 프로젝트의 "examples/functions/hybrid_quant" 디렉토리에는 하이브리드 양자화 예제가 있습니다. 사용자는 이 예제를 참고하여 하이브리드 양자화를 수행할 수 있습니다.

1.3.3 Automatic Hybrid Quantization

하이브리드 양자화 사용을 간소화하고 모델 오버플로우 문제를 해결하기 위해 RKNN-Toolkit2는 자동 하이브리드 양자화 기능을 제공합니다. 사용자는 auto_hybrid 인터페이스를 사용하여 모델 양자화 과정에서 유클리드 거리 및 코사인 거리 임계값을 기반으로 하이브리드 양자화 모델을 자동으로 조정할 수 있습니다. 양자화되지 않은 모델의 경우, 모델의 각 레이어를 FP16 오버플로우 여부를 검사하고 오버플로우가 발생한 레이어를 INT16 계산으로 변환합니다.

현재 자동 하이브리드 양자화 기능은 다음과 같은 사용 시나리오를 지원합니다:

1. 모델 양자화 과정에서 하이브리드 양자화 모델은 유클리드 거리 및 코사인 거리 임계값에 따라 자동으로 조정될 수 있습니다. 현재는 int8 양자화에서 fp16으로의 자동 조정만 지원됩니다.

```
ret = rknn.build(do_quantization=True, dataset='./dataset.txt', auto_hybrid=True)
```

코사인 거리와 유클리드 거리의 임계값은 config 인터페이스에서 auto_hybrid_cos_thresh 와 auto_hybrid_euc_thresh를 사용하여 구성할 수 있습니다. 코사인 거리 기본값은 0.98이며, 유클리드 거리 기본값은 None으로 설정되어 있어 활성화되지 않습니다. 즉, 코사인 거리 정밀도가 임계값보다 작게 지정된 연산자만 기본적으로 fp16으로 변환되며, 이 시점에서 유클리드 거리 임계값은 비활성화됩니다. 유클리드 거리 임계값이 활성화되면, 코사인 거리 임계값보다 작거나 유클리드 거리 임계값보다 큰 연산자는 fp16으로 변환됩니다.

2. 비정량화 모델의 경우, 모델의 각 레이어를 fp16 오버플로우 여부를 검사하고 오버플로우가 발생한 레이어를 int16 연산으로 변환합니다. 이 때, 각 레이어의 수치 범위를 계산하기 위해 데이터셋을 제공해야 합니다.

```
ret = rknn.build(do_quantization=False, dataset='./dataset.txt', auto_hybrid=True)
```

주어진 입력에 따라 각 레이어별로 최대값과 최소값을 계산합니다. 최대값 또는 최소값이 fp16의 상한값 또는 하한값을 초과할 경우 해당 레이어의 연산자는 int16으로 변환됩니다.

1.4 Quantization-Aware Training (QAT)

1.4.1 Introduction to QAT

양자화 인식 훈련(QAT)은 저비트 양자화 시 정확도 손실 문제를 해결하기 위한 훈련 방법입니다. 저비트 양자화는 부동소수점 수에서 고정소수점 수로 값 범위를 변환하는 과정에서 정밀도 손실이 발생하기 때문에 정확도 손실을 초래할 수 있습니다. QAT 훈련 중에는 양자화 오차가 훈련 손실 함수에 포함되어 양자화 매개변수를 가진 모델을 훈련할 수 있습니다.

RKNN-Toolkit2에서 제공하는 사후 훈련 양자화(PTQ)와 대비하여, 이 두 양자화 방법의 특징은 다음과 같습니다:

양자화 방법	원본 프레임워크를 기반으로 한 2차 학습	Dataset	가중치 파라미터 조정	손실 함수	Performance
훈련 후 양자화 (PTQ)	No	라벨이 없는 소량 데이터	No	상관 없음	최적
양자화 인식 훈련 (QAT)	Yes	완전한 훈련 데이터셋	Yes	양자화 손실은 훈련 손실 함수에 포함됩니다.	QAT를 지원하지 않는 연산자가 있는 경우, 성능은 PTQ보다 약간 낮습니다.

1.4.2 QAT Principle

QAT 훈련 중에는 모든 가중치가 저장되고 계산이 부동소수점 형식으로 수행됩니다. 이는 역전파가 정상적으로 작동하고 모델이 효과적으로 훈련될 수 있도록 보장합니다. 부동소수점 모델 훈련과 달리 QAT는 모델의 양자화 가능한 위치에 FakeQuantize 모듈을 삽입하여 부동소수점 숫자를 고정소수점 숫자로 변환할 때 발생하는 정밀도 손실을 시뮬레이션합니다. 이로써 손실 함수가 양자화를 인식하고 최적화할 수 있어, 모델이 고정소수점 모델로 변환될 때 정확한 추론 결과를 보장합니다.
QAT 훈련은 주요 추론 프레임워크에서 널리 사용되고 지원됩니다. 자세한 사용 방법은 아래 링크를 참조하세요.

Pytorch - https://p y torch.org/blog/quantization-in-practice/#quantization-aware-training-qat

Paddle - https://paddleslim.readthedocs.io/zh-cn/develop/api_cn/dygraph/quanter/qat.html

Tensorflow - https://www.tensorflow.org/model_optimization/guide/quantization/training

1.4.3 QAT Usage Guidelines

QAT를 사용하려면 추가적인 훈련 코드가 필요하며, 일부 오픈소스 저장소의 기능과 충돌이 발생할 수 있으므로, 다음 두 조건이 모두 충족될 경우 QAT 사용을 고려하는 것이 좋습니다:
제7장을 참조하여 양자화 정확도 분석을 수행하여 RKNN의 PTQ 기능이 정확도 요구사항을 충족하지 않는지 확인하십시오.
제6.3장을 참조하여 하이브리드 양자화를 탐색하고 RKNN의 하이브리드 양자화 기능이 정확도와 성능 요구사항을 충족하지 않는지 확인하십시오.

1.4.4 QAT Implementation Example and Configuration Instructions

다음은 다양한 프레임워크에서 QAT 기능에 대한 문서입니다. 실제 사용 방법은 공식 문서를 참고해 주시기 바랍니다.

PyTorch: https://pytorch.org/docs/stable/quantization.html (PyTorch에는 여러 개의 양자화 인터페이스 세트가 있습니다. RKNN은 현재 FX 인터페이스에서 생성된 양자화 모델을 지원하며, 특히 prepare_qat_fx 인터페이스와 관련된 인터페이스를 지원합니다.)

Paddle: https://www.paddlepaddle.org.cn/tutorials/projectdetail/3949129#anchor-14

TensorFlow: https://www.tensorflow.org/model_optimization/guide/quantization/training

PyTorch를 예시로 들어 QAT 구현 과정과 중요한 고려 사항을 설명합니다.

```
# for 1.10 <= torch <= 1.13
import torch
class M(torch.nn.Module):
 def ___init___(self):
 super().___init___()
 self.conv = torch.nn.Conv(3, 8, 3, 1)

 def forward(self, x):
 x = self.conv(x)
 return x

# initialize a floating point model
float_model = M().train()

from torch.quantization import quantize_fx, QConfig, FakeQuantize, MovingAverageMinMaxObserver,
MovingAveragePerChannelMinMaxObserver

qconfig = QConfig(activation=FakeQuantize.with_args(observer=
              MovingAverageMinMaxObserver,
              quant_min=0,
              quant_max=255,
              reduce_range=False),   # reudece_range -> the default is True

 weight=FakeQuantize.with_args(observer=
          MovingAveragePerChannelMinMaxObserver,
          quant_min=-128,
          quant_max=127,
          dtype=torch.qint8,
          qscheme=torch.per_channel_affine,
          #qscheme->the default is torch.per_channel_symmetric
          reduce_range=False))
qconfig_dict = {"": qconfig}
model_qat = quantize_fx.prepare_qat_fx(float_model, qconfig_dict)

# define the training loop for quantization aware training
def train_loop(model, train_data):
 model.train()
 for image, target in data_loader:
   ...
# Run training
train_loop(model_qat, train_loop)

model_qat = quantize_fx.convert_fx(model_qat)
```

위 코드 흐름에서 RKNN 하드웨어를 위한 qconfig 구성에 대한 조정 사항을 제외하고, 모든 다른 작업은 공식 코드에서 제공된 지침에 따라 구현되었습니다. qconfig에는 두 가지 주요 변경 사항이 있습니다:

활성화 양자화 구성에서 reduce_range를 False로 지정합니다. reduce_range가 False일 경우 효과적인 양자화 범위는 -128부터 127까지입니다. reduce_range가 True일 경우 효과적인 양자화 범위는 -64부터 63까지로, 이는 양자화 성능이 저하됩니다. RKNN 하드웨어는 reduce_range를 False로 지원합니다.

가중치 양자화 구성은 qscheme을 torch.per_channel_affine으로 지정합니다. 기본값인 torch.per_channel_symmetric은 zero_point를 0으로 고정하지만, RKNN 하드웨어는 0이 아닌 zero_point를 지원합니다. 따라서 torch.per_channel_affine이 선택되었습니다.

1.4.5 Supported Operators in QAT

PyTorch를 예로 들면, QAT 프로세스 동안 가중치 매개변수를 가진 Conv 및 Linear 연산자는 먼저 QAT 규칙에 따라 양자화됩니다. 그 다음 다른 연산자들이 검토됩니다. 만약 해당 연산자가 일반 양자화 규칙을 충족한다면 일반 양자화 방식으로 양자화됩니다. 만약 연산자가 QAT 및 일반 양자화 규칙을 모두 충족하지 않는 경우, 해당 연산자는 FP32 계산 규칙을 유지합니다.

다양한 프레임워크는 서로 다른 수준의 지원을 제공합니다. 사용자는 사용 중인 프레임워크와 버전별로 지원되는 양자화 연산자에 대한 자세한 정보를 확인하려면 다음 링크를 참조할 수 있습니다:

Pytorch: https://github.com/pytorch/pytorch/blob/main/torch/ao/quantization/quantization_mappings.py

Paddle: https://github.com/PaddlePaddle/Paddle/blob/86df789a567f1285101c57b6e3ada4b952c58f48/python/paddle/quantization/config.py

Tensorflow: https://www.tensorflow.org/model_optimization/api_docs/python/tfmot/quantization/keras/QuantizeConfig

1.4.6 Handling of Floating-Point Operators in QAT Model

QAT 모델에서 연산자가 양자화될 수 없는 경우 부동소수점 연산이 사용됩니다. 이러한 연산자를 RKNN 모델로 변환할 때 고려해야 할 두 가지 시나리오가 있습니다.

1. 앞뒤의 모든 연산자는 양자화 가능합니다:

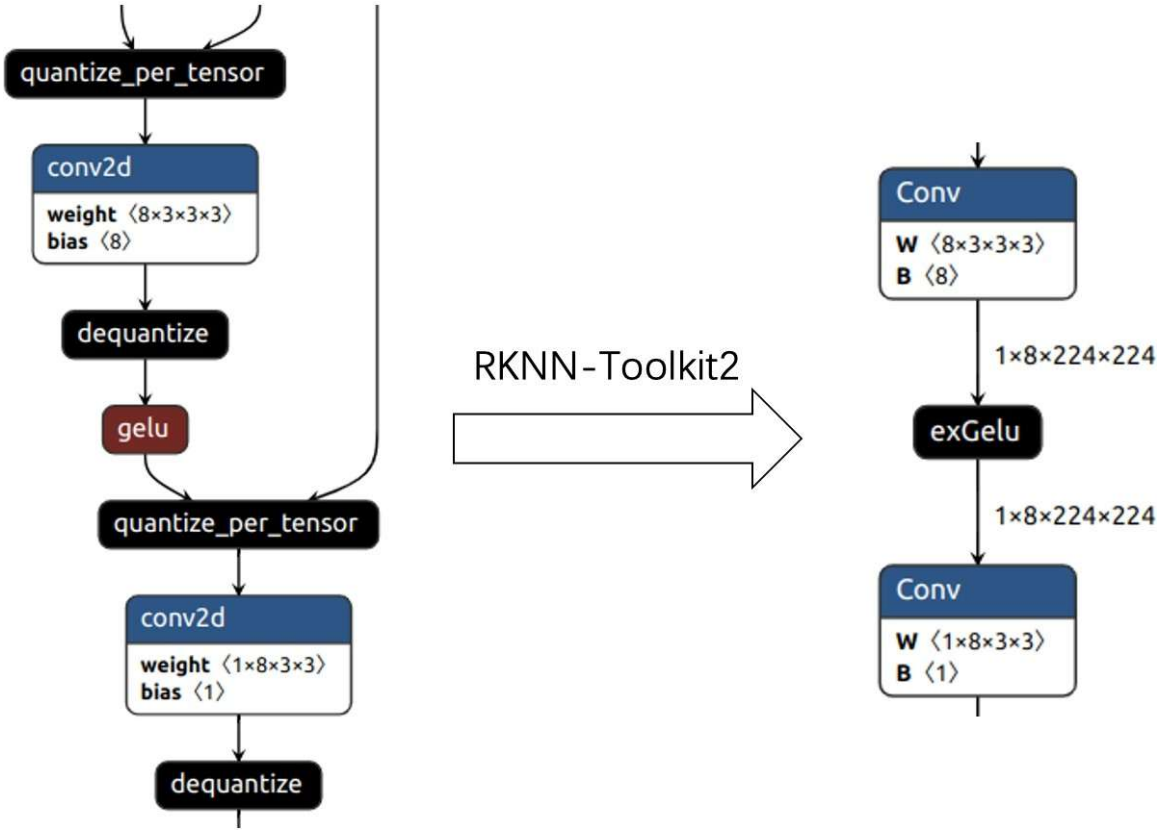

Figure 6-5 QAT OP is preceded and followed by quantifiable operators

위 그림의 왼쪽에 표시된 것처럼, 원본 모델의 gelu 연산자는 부동소수점 연산자이며, 앞뒤에 위치한 conv2d 연산자는 양자화되어 있습니다. RKNN-Toolkit2를 사용하여 모델을 로드할 때, 두 양자화 연산자 사이의 부동소수점 연산자는 정확도 손실 없이 추론 성능을 향상시키기 위해 양자화 연산자로 변환됩니다. RKNN 모델로 변환된 후의 구조는 위 그림의 오른쪽에 표시된 것과 같습니다.

2 비정량화 가능한 연산자가 앞에 또는 뒤에 존재합니다:

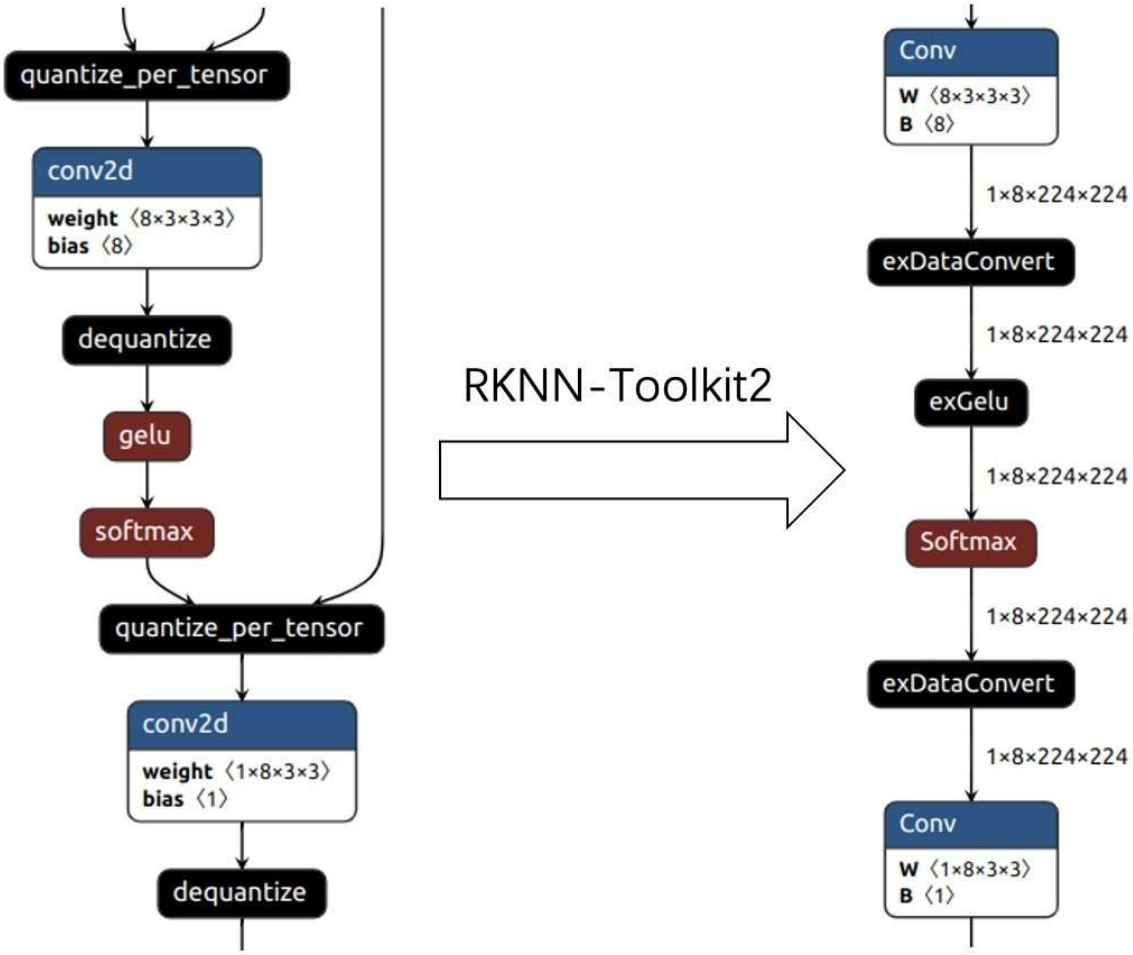

Figure 6-6 QAT OP is preceded and followed by non-quantifiable operators

위 그림의 왼쪽에 표시된 것처럼, 원본 모델의 gelu 및 softmax 연산자는 부동소수점 연산자이며, 앞뒤에 위치한 conv2d 연산자는 양자화되어 있습니다. RKNN-Toolkit2로 모델을 로드할 때, gelu 및 softmax에 양자화 매개변수가 결여되어 있으므로 이들은 부동소수점 형식으로 유지됩니다. RKNN 모델로 변환된 후, 위의 오른쪽 그림에서 볼 수 있듯이 exGelu 앞에 양자화 해제 연산자 exDataConvert가 삽입되고, Softmax 뒤에 양자화 연산자 exDataConvert가 삽입됩니다.
이 삽입된 양자화 및 양자화 해제 연산자는 추가적인 지연 시간을 유발합니다.

2.1.1 Summary of QAT Experience

1. QAT 설정

QAT 훈련은 다양한 하드웨어의 특성에 따라 더 나은 결과를 얻기 위해 구성 설정을 조정하는 것이 자주 필요합니다. RKNPU의 경우, qconfig 매개변수를 구성하기 위해 제6.4.4절의 코드 지침을 참고하는 것이 권장됩니다.

2. 모델 내 저장된 양자화 매개변수의 조정

예를 들어, 시그모이드 함수에서 모델 내 시그모이드 연산자가 기록한 양자화 매개변수는 추론 시 실제로 사용되는 양자화 매개변수와 다를 수 있습니다. 공식 코드(https://github.com/pytorch/pytorch/blob/main/aten/src/ATen/native/quantized/cpu/qsigmoid.cpp)에서 시그모이드 함수의 양자화 매개변수는 추론 중에 조정되며, min을 0으로, max 1로 한다.

3. 이 현상을 해결하기 위해 RKNN-Toolkit2는 모델 변환 단계에서 이러한 연산자의 양자화 매개변수를 조정하여 추론 결과가 원본 PyTorch 추론 결과에 더 가깝게 만들 것입니다.