양자화 분석 및 시도 보고서

1. Aiware에서의 기법

* Scale(Symmetric) quantization
* Per-tensor quantization
* Quantization Process
* MinMax
* Activation Clipping
* Bias Correction
* Cross Layer Equalization

1. 양자화 시도 과정

* 동적 양자화 시도
* weight 양자화 시도
* 정적 양자화 시도
* Yolo\_model.export()
* Onnxruntime

1. 결론
2. 향후 계획
3. Aiware에서의 기법

* Scale(Symmetric) quantization

우선 양자화 분포에는 두 종류가 있는데, 바로 Unform 분포와 Non-uniform분포가 있다. Uniform 분포는 매핑되는 값의 분포가 동일하게 된 것을 말한다. 반대로 Non-uniform 분포는 매핑되는 값의 분포가 동일하지 않은 것을 말한다.

Uniform quantization할 때 scaling factor(실수 값과 정수 값의 변환비율)이 중요하다. scaling factor는 실수값 범위를 정수값 범위로 균등하게 압축해 주는 역할을 한다. scaling factor를 정의하기 위해 clipping range[a ,b]를 정해야 한다.

가장 쉬운 방법은 min/max를 기준으로 하는 것이다. a 가 max, b 가 min이라고 할 때, a=b 가 symmetric, a!=b면 asymmetric이라고 한다. 결국 clipping range(범위 바깥의 값들이 최대, 최소 값으로 매핑)가 대칭 혹은 비대칭의 문제이다.

aiWare Studio에서는 Scale(Symmetric) Quantizatoin 방식을 사용한다. Scale Quantization은 z-point(양자화에서 실수 값을 정수 값으로 변환할 때 사용하는 기준 값 – 실수 값0이 정수 값으로 매핑되었을 때 어떤 위치에 있는지)을 사용하지 않고 값의 범위를 대칭적으로 압축한다.

* Per-tensor quantization

양자화에서 적용 범위에 따라 2가지 종류가 있다. Per-channel quantization과 per-tensor quantization이다. 이중 aiWare Studio는 per-tensor quantization을 사용한다. 우선 per-channel quantization은 모든 convolution kernel이 다른 clipping range를 가진다. 이 방법의 장점은 채널별로 scaling factor와 z-point를 반영하여 양자화 하여 모델의 정확도가 높은 것이고, 단점은 모든 채널이 scaling factor와 z-point를 가지고 있어 모델이 무거워지고 엣지 디바이스에서의 호환성이 상대적으로 좋지 않다.

per-tensor quantization은 모든 tensor에 대해 하나의 scaling factor, z-point를 반영하여 양자화를 하는 것이다. 이 방법의 장점은 위의 per-channel quantization과 반대로 각각의 tensor, 혹은 channel별로 반영하여 양자화 하지 않기 때문에 연산이 단순해져 모델이 가벼워지고 엣지 디바이스에서의 호환성이 상대적으로 좋다. 하지만 per-channel quantization에 비해 정확도가 떨어진다는 단점이 있다.

* Quantization Process

aiWare Studio에서의 NPU는 int8 data type을 지원한다. 따라서 기존의 학습 모델에 사용하던 float32를 int8로 양자화하는 과정이 필요하다. 첫 번째로, 최댓값과 최솟값의 절댓값을 비교하여 더 큰 값을 boundary value로 설정한다. 이때 Min/Max는 aiWare에서는 import된 모델 NNEF의 graph.stats이라는 파일을 참조한다.

각 boundary 값의 절댓값 보다 큰 2의 거듭제곱을 기준으로 boundary 2차 설정한다. 예를 들면 최댓값의 절댓값이 최솟값보다 높은 17이라고 했을 때 2차설정되는 값은 32이다. 이 값을 기준으로 int8 boundary에 매핑을 수행한다. 3.56이라는 값을 기준으로 하여 예를 들면 round(3.56 \* (256/(32\*2))) = round(3.56 \* (2^8/2^6)) = round(14.24) = 14 의 값으로 mapping 된다.

모델 내에서 각 레이어가 취급하는 bit 범위가 다를 수 있는데, 이 때문에 Dequantize(역 양자화)가 필요하다. 역 양자화는 그것 이외에도 양자화 과정에서 실수 값을 정수 범위로 압축하면서 손실된 정보를 최대한 회복한다. 앞에서 매핑된 값을 반대로 14 \* (32 \* 2 / 256) = 14 \* (1/4) = 3.5이다. 복원된 값은 양자화 과정에서 반올림 연산자를 사용한 이유로 원래의 실수 값과 완전히 동일하지는 않다.

* MinMax

aiWare Studio는 네가지 종류의 양자화 메소드를 제공하는데, 각 메소드에 따라 weight 또는 activation에서 서로 다른 statistic(통계값)을 추출하거나 재조정한다. Import한 모델의 network의 상태에 따라 다양한 메소드를 적용하거나 조합할 수 있다.

그 중 첫번째로 다룰 메소드는 바로 MinMax메소드이다. MinMax메소드는 weight 또는 activation 값의 범위(Min, Max)를 사용하여 값을 양자화하는 메소드이다. 이 방법은 간단하고 효율적인 방법이다. 값의 범위만을 기준으로 양자화를 수행한다. 때문에 다른 방법에 비해 메모리 사용량이 적다. 하지만 입력 데이터가 균등하게 분포되어 있지 않으면, 혹은 극단적으로 분포된 경우에 있어 정보손실이 날 수 있어 성능이 저하될 수 있는 단점이 있다.

* Activation clipping

Activation clipping은 활성화 값의 통계 값을 조정하는 방식이다. 실수 값의 활성화 값에 대해 histogram을 생성 후 왼쪽 끝을 기준으로 8bit boundary의 histogram을 생성 후 그 값의 분포를 분석하여 Clipping경계를 결정한다. Clipping을 통해 활성화 값의 상한 값과 하한 값을 설정하고 그 범위를 넘어서는 값은 상한 값과 하한 값으로 고정한다. 활성화 값을 양자화 할 때 값 범위를 좁혀 손실되는 값을 줄인다. 히스토그램에서 왼쪽 혹은 오른쪽 끝의 활성화 값을 기준으로 clipping 경계를 설정하여 불필요하게 큰 값이 모델 성능에 끼치는 영향을 줄인다. 하지만 activation clipping의 범위가 잘못 설정되면 중요한 정보가 손실되어 모델 성능이 떨어질 수 있다.

* Bias Correction

Bias Correction은 Bias tensor의 값을 조정하는 방식인데. 역 양자화된 출력 값에 대해서 실수 값의 출력 값에 오차가 추가된 것으로 정의하는데 두 출력 간의 오차를 계산하여 bias에 injection하는 방식으로 bias를 조정한다.

오차가 bias 텐서에 추가되는 과정에서 오차는 각 layer에서 실수 값 출력과 양자화된 출력의 차이를 계산하여 해당 layer의 bias 값에 더해주는 방식으로 보정한다. 이 방법은 양자화된 모델이 원본 모델과 더 비슷하게 출력을 내도록 도와 양자화 후 발생하는 성능 저하를 보완한다. 하지만 오차가 지나치게 커지면, bias 텐서만으로 보정하지 못 할 수 있고, 모든 layer에서 오차를 계산하고 보정하기 때문에 양자화 과정이 더 복잡해질 수 있다.

* Cross Layer Equalization

Cross Layer Equalization은 가중치 텐서의 값을 조정하는 방식이다. Depthwise convolution의 경우 연산 특성상 weight와 channel간 서로 다른 통계값을 가지고 있을 가능성이 있다. 이 때 양자화시 범위가 큰 channel의 통계값이 반영되어 상대적으로 작은 범위를 가진 channel에 대해서 정보 손실이 발생한다. 이때 각 group에 대해 scale을 유지하게 해주는 대각 행렬을 구하여 나누고 다음 layer에 영향을 끼치는 weight에 곱하는 방식으로 weight을 flatten시킨다.

예를 들면 연속된 두 레이어 x, y가 있다고 할 때 x의 가중치가 크고, y의 가중치가 작으면 Cross Layer Equalization은 x의 가중치를 줄이고 y의 가중치를 증가시키는 방식으로 전체 네트워크의 출력 분포를 유지한다. 이 방법의 장점은 int8로 양자화할 때 값의 범위가 너무 크거나 작아 발생하는 양자화 quantization error를 줄이고 레이어간 스케일 불일치로 인한 학습 불안정성도 감소한다. 하지만 CLE는 스케일 균형을 맞춰 양자화 성능을 최적화 하는 방법 답게 ReLU와 같은 비선형성 활성화 함수에 잘 어울린다. 그래서 SiLU가 기본인 yolo에는 잘 맞지 않다고 생각한다.

1. 양자화 시도 과정

* 동적 양자화 시도

첫 번째로, 학습한 가중치 파일을 onnx 파일로 변환 후 동적 양자화를 시도했다. 동적 양자화는 각 feature map에 대하여 range를 동적으로 계산하는데, 이는 각 feature map의 통계값의 실시간 연산을 요구하므로 오버헤드가 매우 크다. 그런 만큼 높은 정확도를 얻을 수 있다. 하지만 모델이 무거워지고 연산처리에 시간이 더 걸린다.

동적 양자화 후 aiWare Studio에 import를 시도했지만 DynamicQuantizeLinear라는 연산자는 지원하지 않는다는 에러 메시지를 확인했다. 당시에는 몰랐지만 후에 aiWare Studio는 양자화된 모델을 import하는게 아닌 양자화 요청상태의 모델을 import하여 aiWare 내부에서 양자화 하는 것을 알게 되었다.

* Weight 파일 양자화 시도

위의 방법이 실패한 후 pytorch에서 제공하는 quantize\_dynamic이란 모듈을 이용해 .pt(weight)파일로 양자화를 한 후에 onnx파일로 내보내는 시도를 했다. quantize\_dynamic 함수 내에 모델을 로드하고 양자화할 종류를 정한 뒤 양자화의 type을 in8로 설정하였는데, 더미 데이터가 필요하다는 것을 알았다. 그래서 yolo모델에서 summary한 것처럼 (1, 3, 640, 640)형태의 더미 데이터를 선언 후 양자화 하였다.

위에서 언급하였지만, aiWare Studio에서는 양자화된 모델이 아닌 요청상태의 모델을 받아 양자화하기 때문에 당연히 import하는데 실패하였다. 또한 동적 양자화로 시도하였기 때문에 올라가지 않는 것이 당연했다. 이 시도 이후 무작정 변환하고 다른 기법을 찾아보려 했던 생각이 틀렸다는 것을 깨닫고 다른 기법들을 찾아보기 시작했다.

* 정적 양자화 시도

우선 동적 양자화를 알아보았을 때 다른 calibration range방법인 정적 양자화를 시도했다. 동적 양자화가 각 feature map에 실시간 연산을 요구했다면 정적 양자화는 미리 계산된 clipping range를 추론에 사용한다. 이에 따라 추가적인 연산을 요구하지 않으므로 오버헤드가 발생하지 않아 가볍고 빠르다. 하지만 동적 양자화와 비교했을 때 정확도가 더 낮다.

정적 양자화를 수행하기 위해서는 캘리브레이션 데이터가 필요하다. 이 데이터는 모델이 학습된 데이터의 분포를 반영하고 모델의 입력 형태에 맞춰야 한다. 캘리브레이션 데이터 셋은 학습모델의 valid 데이터 셋 일부를 사용했다. 그런데 추론만 되어야할 모델이 학습 모드로 돌아가서 포기하였다. 이 부분도 좀 지나서 다시 조사해봤더니 torch.no\_Grad()로 gradient계산을 비활성화 시켜서 gradient를 추적하지 않게 해줘야 한다는 것을 알았다.

* Yolo\_model.export()

다양한 시도를 하다가 ultralytics 공식 문서를 보게 되었는데 YOLO모델을 선언하고 export(format=’onnx’, int8=’True’) 등등 자체 함수에서 양자화 및 최적화, opset버전, 원하는 파일 형태로 format하는 인자들을 지원한다는 것을 알았다. 이 자체 함수를 이용하여 많은 모델을 실험하였다.

export()함수를 알고나서 본격적으로 양자화 종류, 방법 등 여러 자료를 조사하기 시작했다. 그 결과 aiWare Studio에서 지원하는 연산자나 방식을 참고하여 ultralytics 자체함수인 export()가 아닌 다른 방법을 찾기 시작했다. 다른 다양한 방법을 찾으면 aiWare Studio에 올리기 전에 더 다양한 기법을 적용할 수 있을 것이라고 생각했다.

* Onnxruntime

onnx는 onnxruntime이라는 모듈을 지원하는데 onnxruntime은 ONNX 모델을 실행하기 위한 범용적인 실행 엔진이다. 먼저 ultralytics의 자체함수인 export()로 가중치 파일을 onnx로 변환한 후 onnx모델을 로드한다. 앞서 말했듯이 동적 양자화는 지원하지 않으니 정적 양자화로의 변환을 시도했다. 정적 양자화를 하기위해 캘리브레이션 데이터의 경로를 선언했고, 이미지 데이터 셋을 모델에 맞게 조정했다. 그런 다음 이미지 데이터 셋을 quantize\_static모델로 불러와서 양자화를 진행하고 양자화 연산자를 제거하여 양자화 요청상태로 변환하였다.

quantize\_static함수 내의 인자에서 calibrate\_method가 있다는 것을 확인하였고, CalibrationMethod를 보니 Default인 MinMax말고도 세 가지가 더 있다는 것을 확인하였다. 간단히 설명하자면

* Entropy : 분포가 넓거나 극단차가 있는 데이터에 유리
* Percentile : 특정 백분위수로 양자화 범위를 결정하여, 이상치의 영향을 줄이고 정밀도를 높임
* Distribution : 데이터의 분포를 기준으로 양자화 범위를 더 조정

이 세가지 방법들은 현재 필자의 노트북 메모리보다 더 많은 메모리를 사용하는 것을 확인했다. 그래서 MinMax로 양자화 한 후 aiWare Studio에 import를 시도하였다.

하지만 aiWare Studio에서 export()함수로 양자화 요청상태로 import한 모델과 비교하여 성능변화가 없는 것을 확인했고 npu상의 mAP도 동일한 것을 확인했다. 그래서 원인을 알아보기 위해 export()를 들여다보니 export()함수 자체가 onnxruntime을 이용하여 onnx변환 및 양자화 요청상태로 변환하는 것을 확인하였다.

1. 결론

지금까지 다양한 방법으로 시도했지만, export와 onnxruntime말고는 마땅한 양자화 변환 방법을 찾지 못 하였다. 하지만 다양한 시도 과정에서 얻은 지식을 기반으로 지금까지 실험한 모델들의 데이터 값 분포, 활성화 값 분포 등등 각각 모델들의 특징들을 추출하여 aiWare Studio의 quantization method, 각각 MinMax, Activation Clipping, Bias Correction, Cross Layer Equalization을 각 기법의 특성에 맞는 모델에 적용시킨다. 그렇게 함으로써 모델 양자화 과정에서 생기는 가중치 양자화 손실을 줄여 npu에서의 연산 양자화 손실을 더욱 줄일 수 있다.

이러한 과정에서 중요한 점은 양자화가 단순한 속도 최적화만 하는 것이 아니라는 것이다. 모델의 효율성과 정확도를 동시에 고려하여 모델의 특징에 맞는 양자화 기법을 선택하는 것이 중요하며, 각 양자화 방법이 모델을 양자화할 때 발생하는 가중치 양자화 손실을 줄이는 방향으로 최적의 조합을 찾아가는 것이 핵심이다.

1. 향후 계획

aiWare Studio의 설치경로 내에 각각 quantization method 폴더가 있는 것을 확인하였다. 이를 바탕으로 각각 quantization method의 파이썬 스크립트를 분석하고, 어떤 특징이 있는지 특정한다. 이후 연산 양자화 손실에 관해서 실험한 각각의 모델들에 대한 특징을 추출하고 aiWare Studio에서 quantization method를 각각 혹은 조합해서 적용시켜 npu에서 성능을 비교해 최선의 모델을 찾아낸다. 고려할 점은 현재 aiWare Studio 버전에서 yolov8 버전에 대한 quantization method 지원을 하지 않기 때문에 yolov5 버전을 기준점으로 잡아 실험할 것이다.