**연산 양자화 분석 보고서**

1. **서론**
   1. 배경
      1. Yolo 모델의 연산 양자화 필요성  
         : 모델 네트워크 구조를 적절히 변경하거나 수정하여 GPU에서 성능이 높게 나와도, NPU에서 성능이 높게 나올 것이라고 확신할 수 없다. GPU 에서의 성능이 중요하긴 하지만, NPU 상에서 성능이 떨어지게 된다면 real time & on-device 입장에서 잘 설계한 모델이라고 할 수 없다.   
         GPU는 32bit 부동소수점 연산에 최적화되어 있지만 NPU에서는 8bit 고정소수점 연산으로 이루어진다. 따라서 GPU에서의 성능이 NPU에서 최대한 손실되지 않게 하기 위해서 연산 양자화 손실을 최소화하는 방법에 대해 연구해야 한다.  
         실험에 앞서, 용어의 통일성을 위해 본 실험에서 GPU 성능 대비 NPU 성능의 감소를 **연산 양자화 손실**, NPU/GPU 성능 비율을 보존율이라 하겠다.
   2. 목적
      1. Yolo 모델의 레이어 설계와 모듈 선택이 연산 양자화 손실에 미치는 영향을 정량적으로 분석하고, 성능과 효율성의 균형을 유지할 수 있는 최적의 구조 **조합**을 도출하는데 목적을 둔다.  
         최종적으로 본 실험의 결과를 반영하여 연산 양자화 손실이 적은 최적의 모델을 선정할 수 있도록 한다.
   3. 실험 준비
      1. 실험 항목
         1. BackBone의 첫 레이어(Conv)의 kernel\_size 차이 : (6, 2, 2, 2) vs (3, 2).
         2. Bottleneck 반복 수에 따른 성능 분석 : (2, 4, 4, 2) vs (6, 8, 8, 6).
         3. Shortcut 활성화에 따른 C2f, C3, C3Ghost 비교.
         4. Bottleneck 반복 수와 C2f, C3, C3Ghost의 최적 조합 비교
         5. Conv와 GhostConv의 성능 차이.
         6. Conv, GhostConv와 C2f, C3, C3Ghost의 조합 비교
         7. 채널 수에 따른 손실 비교
      2. 평가 지표
         1. Parameter 수, NPU/GPU 비율, NPU 성능, GPU 성능, FPS
2. **본론**
   1. 실험 설계
      1. 실험 방법
         1. 동일한 데이터로 학습한 모델에 대해 동일한 입력 데이터를 사용하여 실험 1모델 별 성능 측정.
            1. 학습 데이터 : 제공받은 train set과 validation set을 병합
            2. 입력 데이터 : 제공받은 test est
         2. 실험할 주제 외의 항목은 통제.
   2. 실험 결과 및 분석
      1. 실험 1 Conv 첫 레이어 kernel size에 따른 성능  
         1. 비교 모델 : v8n\_org\_tv vs v8n\_k6\_tv
         2. 모델 설명 :
            1. V8n\_org\_tv : 기본 yolov8n 모델
            2. V8n\_k6\_tv : 기본 yolov8n 모델의 첫 Conv 레이어의 크기를 (3, 2) 에서 (6, 2, 2, 2)로 변경한 모델
         3. 가설   
            : kernel size ->   
            Convolution의 kernel size가 커질수록 연산 복잡도가 증가하여 NPU에서 양자화 손실이 커질 가능성이 있다. Kernel size가 커지면 필터가 더 넓은 범위의 정보를 학습할 수 있지만, 각 연산에서 사용되는 값의 범위가 넓어져 양자화 시 표현할 수 없는 정보가 증가할 수 있다[1].   
            따라서 커널 크기가 큰 모델의 연산 양자화 손실이 클 것이다.
         4. 결과 :
         5. 분석 :
      2. 실험 2 Bottleneck 반복 수에 따른 연산 양자화 손실  
         1. 비교 모델 :
            1. v8n\_2442 vs v8n\_6886
            2. v5n\_2442 vs v5n\_6886
         2. 모델 설명 :
            1. v8n\_2442 / v8n\_6886 : 기본 yolov8n 모델에 bottleneck 반복 수 2, 4, 4, 2를 설정한 모델
            2. v5n\_2442 / v5n\_6886 : 기본 yolov5n 모델에 bottleneck 반복 수 2, 4, 4, 2를 설정한 모델
         3. 가설   
            : **Bottleneck** 반복 수가 증가하면 모델의 연산 복잡도가 늘어나고 레이어 간 연결과 정보의 흐름이 깊어진다. 이는 8bit 연산에서 양자화 손실을 증가시킬 가능성이 있다[2]. 따라서 A, B 두 경우 모두 6886 모델이 2442 모델보다 보존율이 떨어질 것이다.
         4. 결과 :
         5. 분석 :
      3. 실험 3 shortcut 활성화에 따른 C2f, C3, C3Ghost 성능 비교  
         1. 비교 모델 :
         2. 모델 설명 :
            1. a
            2. b
         3. 가설 :
            1. Shortcut=True(Backbone)  
               : shortcut=True 설정에서 원본 입력이 마지막에 concatenate 되는데 정보 간 차이가 커질 가능성이 높아 손실이 많아질 것이다.  
               - C2f의 경우 concat하는 채널 개수가 많아 연산량이 많기 때문에 손실이 클 것이다[3].   
               - GhostBottleneck은 Bottleneck보다 효율적인 연산량 감소를 목표로 하지만, 더 많은 연산 단계를 가지는 특성상 양자화 손실이 커질 가능성이 있다[4].   
               - 따라서 손실량은 C2f > C3Ghost > C3 순서로 나타날 것이다.
            2. Short=False(Neck)  
               : shortcut=False 설정에서는 원본 입력이 포함되지 않더라도, shortcut=True일 때와 마찬가지로 손실량은 C2f > C3Ghost > C3 순서로 나타날 것이다.
         4. 결과 :
         5. 분석 :
      4. 실험 4 Bottleneck 반복 수와 C2f, C3, C3Ghost의 최적 조합 비교  
         1. 비교 모델 :
         2. 모델 설명 :
            1. C2f\_2442, C3\_2442, C3Ghost\_2442
            2. C2f\_6886, C3\_6886, C3Ghost\_6886
         3. 가설 :
            1. Bottleneck 반복 수가 증가하면 레이어 간 연결과 정보 흐름이 깊어진다. 하지만 이는 8bit 양자화 손실을 일으키며, 특히 더 많은 레이어와 concat 연산을 가진 구조에서 손실이 더 커질 것이다.   
               - 2442 반복에서는 연산 복잡도가 낮아 구조의 특성이 더 크게 작용하므로, C3 > C3Ghost > C2f 순서로 성능이 좋을 것이다.  
               - 6886 반복에서는 반복 수 증가로 인해 양자화 손실의 영향을 크게 받아, 손실량이 C2f> C3Ghost > C3 순으로 일어날 것이다.
            2. 근거  
               - 실험 2에서 Bottleneck 반복 수가 증가할수록 양자화 손실이 커지는 경향이 관찰되었다.  
               - C2f는 concat 연산 개수가 많고 C3Ghost는 Ghost Module 특성으로 내부 연산 단계가 많아, 반복 수 증가 시 양자화 손실에 더 취약하다.  
               - C3는 상대적으로 단순한 구조로, 반복 수 증가의 영향을 덜 받을 가능성이 있다.  
               따라서 반복 수가 많을 때와 적을 때 모두 구조 특성상 단순한 C3의 연산 양자화 손실이 가장 적고, C2f의 손실이 가장 클 것이다.
         4. 결과 :
         5. 분석 :
      5. 실험 5 Conv와 GhostConv의 성능 차이  
         1. 비교 모델 :
         2. 모델 설명 :
            1. a
            2. b
         3. 가설   
            : GhostConv는 Conv와 다르게 concat 연산이 추가되므로, 8bit 연산에서 정보 손실이 더 클 가능성이 있다..

Conv는 상대적으로 간단한 연산 구조로, 양자화 연산에서의 손실이 적을 것이다.  
따라서 Ghostconv가 Conv보다 연산 양자화 손실이 클 것이다.

* + - 1. 결과 :
      2. 분석 :
    1. 실험 6 Conv, GhostConv와 C2f, C3, C3Ghost의 최적 조합 비교  
       1. 비교 모델 :
       2. 모델 설명 :
          1. a
          2. b
       3. 가설   
          : 위의 실험들을 통해 C2f와 GhostConv에서 가장 연산 양자화 손실이 크다는 것을, C3와 Conv에서 가장 손실이 작다는 것을 알게 되었다. 이를 토대로 조합별 양자화 손실은 아래와 같을 것이다.  
          - 가장 손실이 많은 조합 : C2f + GhostConv  
          - 중간 정도의 손실이 발생하는 조합: C3 + GhostConv, C2f + Conv  
          - 가장 손실이 적은 조합: C3Ghost + Conv
       4. 결과 :
       5. 분석 :
    2. 실험 7 채널 수 증가에 따른 연산 양자화 손실  
       1. 비교 모델 :
       2. 모델 설명 :
          1. a
          2. b
       3. 가설   
          : 채널 수는 모델이 처리할 수 있는 연산 복잡도에 영향을 미친다. 채널 수가 증가하면 더 많은 정보를 연산을 통해 담아야 하므로, 양자화 손실이 더 커질 가능성이 있다. 채널 수가 많은 구성(256, 512, 1024)이 채널 수가 적은 구성(128, 256, 512)보다 더 큰 양자화 손실을 유발할 것이다.  
          따라서 양자화 손실은 채널 수가 더 많은 모델이 클 것이다.
       4. 결과 :
       5. 분석 :

1. **결론**
   1. 주요 결론
      1. C2f vs C3G vs C3
      2. Conv vs GhostConv
      3. Bottleneck
   2. 최적 조합
   3. 향후 연구 방향
2. **참조 문헌**

[1] J. Fernandez-Marques, P. N. Whatmough, A. Mundy, and M. Mattina, "Searching for Winograd-aware Quantized Networks," *Proc. of the 3rd MLSys Conf.*, arXiv:2002.10711, 2020.

[2] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," *arXiv preprint arXiv:1712.05877*, 2017.

[3] <https://github.com/orgs/ultralytics/discussions/15762>

[4] https://pytorch.org/docs/stable/quantization-accuracy-debugging.html