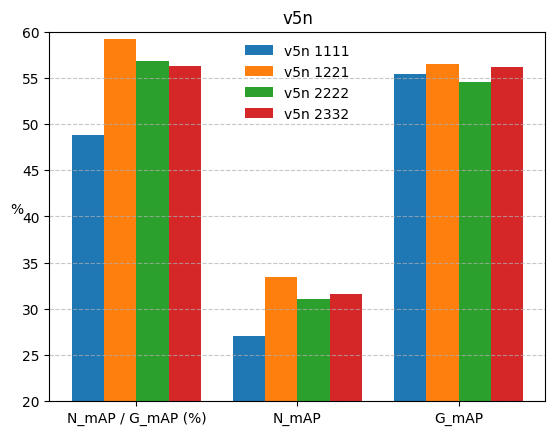
**Bottleneck 반복 수와 보존율 분석 보고서**

1. **서론**
   1. **배경**
      1. Yolo 모델의 구조 형성 방법  
         : 모델의 구조를 선언하는 yolov8의 YAML 파일에서 모델의 사이즈를 결정하는 [depth, width, max\_channel] 인자가 있으며, 각 모듈이 어떻게 적용되는지에 대해 선언한 [from, number, module, args] 형태의 인자가 있다. 이 중 C2f, C3 모듈에 대해서 둘 모두 내부에 Bottleneck 모듈이 존재하는데,  
         **[ Bottleneck 반복 수 = number \* depth(s, n 모델 기준: 0.33) ]**의 형태로 반복 수가 결정된다.
      2. C2f 모듈의 동작 방식  
         \* 보존율 = (NmAP / GmAP) \*100 ( GPU대비 NPU에서 성능이 재현되는 정도)  
         : C2f 모듈은 입력 채널을 두 개로 나눈 후 첫 번째는 shortcut을 통과하고 두 번째는 내부 Bottleneck 모듈에 의해 반복 수 만큼 새롭게 추출된 텐서 그룹들을 만들게 된다. 이 때 텐서 그룹수가 늘어날수록 min/max scale이 다양해지므로 양자화 과정에서 오차를 발생시킬 확률이 늘어난다.  
          따라서 C2f 모듈의 Bottleneck 모듈의 반복수가 늘어나면 최종 concat에서 텐서 그룹수가 늘어나 양자화 오차를 유발해 보존율에 영향을 미치게 된다.
      3. C3 모듈의 동작 방식  
         : C3 모듈은 먼저 입력 채널을 두 개로 나눈 후 첫번째는 cv1을 거친 후 shortcut을 통과한다. 두번째는 cv2를 통해 전달되며, 이후 내부 Bottleneck 모듈을 반복 수만큼 통과하여 특징을 추출한다. 이 때 C2f 모듈과의 차이점은 Bottleneck 반복 수만 늘어나고 최종 concat이 받는 텐서 그룹수는 유지된다는 것이다.
      4. Concat 입력 텐서 그룹수의 양자화 오차 유발  
          결과적으로 v8 모델에서 C2f는 반복수가 늘어나면 concat이 입력받는 텐서 그룹수 증가함과 동시에 Bottleneck 모듈의 적용 횟수가 늘어나므로 변인통제가 되지 않아 입력 텐서 그룹 수만의 영향력을 판단하기 어렵다.  
          따라서 본 연구는 concat 입력 텐서 그룹수가 보존율에 끼치는 영향을 분석하기 위한 선행 연구로써 v5n 모델에서 Bottleneck 반복 수에 따른 보존율과의 상관관계를 먼저 분석한다.
   2. **실험 준비**
      1. 실험 모델
         1. v5n\_1111
         2. v5n\_1221
         3. v5n\_2222
         4. v5n\_2332
   3. **목적**
      1. 실험 1 : Bottleneck 반복 수와 보존율의 상관관계 분석  
         : [모델 1]을 기준으로 C3 내부의 Bottelneck 모듈의 반복 수를 점진적으로 증가시킨 [모델 2], [모델 3], [모델 4]을 보존율의 관점에서 비교하여 두 요소의 상관관계를 분석한다.
   4. **평가 지표**
      1. 모델 성능 : GPU mAP, NPU mAP
      2. 보존율
2. **본론**
   1. **실험 설계**
      1. 실험 방법
         1. 동일한 데이터로 학습한 모델에 대해 동일한 입력 데이터를 사용하여 실험 모델별 성능 측정.
            1. 학습 데이터 : 제공받은 train set과 자체적으로 수집한 data set을 병합
            2. 입력 데이터 : 제공받은 test set
      2. 모델 설계  
         \* 1111 ~ 2332는 Backbone에서 C3/C2f 모듈 내의 Bottleneck 반복 횟수  
         \* **[1.3 Backbone 구조 변경 분석 보고서]**의 [모델 1, 2, 3, 4]와 동일
         1. v5n\_1111
         2. v5n\_1221
         3. v5n\_2222
         4. v5n\_2332
   2. **실험 결과 및 분석**
      1. 실험 1 : Bottleneck 반복 수에 따른 보존율 분석
         1. 실험 항목
            1. 대조군 : v5n\_1111  
               실험군1 : v5n\_1221, 실험군2 : v5n\_2222, 실험군3 : v5n\_2332  
               : v5n 모델에 대해 동일 조건으로 Backbone의 반복 횟수만 변경하여 변인 통제, 가장 낮은 v5n\_1111을 대조군으로 사용한다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | v5n 1111 | **v5n 1221** | v5n 2222 | v5n 2332 |
| 보존율(%) | 48.81 | **59.21** | 56.82 | 56.22 |
| G\_mAP(%) | 55.4 | **56.46** | 54.56 | 56.14 |

* + - 1. 결과  
         - 평가 기준 : 모델 성능, 보존율  
         [그래프 1]
      2. 분석  
         : 위 그래프는 v5n 모델에서 Bottleneck 반복 횟수 차이에 따른 모델별 보존율을 나타낸다.  
          먼저 모델 성능의 관점에서 [그래프 1]의 G\_mAP를 확인했을 때 1111 -> 1221, 2222 -> 2332의 경우, 즉 P3, P4단의 반복수가 증가할 때 성능이 증가하는 것을 확인할 수 있었다. 하지만 1221 -> 2222로 변화할 때 성능이 떨어지는데 이는 **[1.4 P2 적용 및 Backbone 구조 변경 분석 보고서]**에서 얻은 결론과 같이 반복 횟수가 균형 있게 조합되지 않아 생기는 문제로 해석할 수 있다.  
          보존율의 관점에서 [그래프 1]의 보존율을 확인했을 때 1111 -> 1221의 경우에는 큰 폭으로 상승하고, 1221 -> 2222 -> 2332의 경우에는 감소하는 것을 확인했다. v5n\_1111 모델의 경우 절대적인 반복수의 부족으로 N\_mAP 자체가 너무 낮게 나타나기 때문에 v5n\_1221로 변화할 때 보존율이 크게 상승하는 것으로 해석된다.  
          이후의 경우에는 모두 감소하는 경향을 보였다. 따라서 Bottleneck 반복 수가 늘어남에 따라 양자화 오차가 증가하는 것을 확인할 수 있었다.

1. **결론**
   1. **주요 결론**
      1. 실험 1 : Bottleneck 반복 수와 보존율의 상관관계 분석  
         : Bottleneck의 반복 수를 증가시킨 결과 모델의 성능과 관계 없이 보존율이 감소한다는 것을 알 수 있었다. 따라서 이후 C2f 모듈에 대해서 concat의 입력 텐서 그룹수와 보존율과의 연관성에 대한 연구를 진행할 때, Bottleneck의 영향력을 같이 고려해서 분석해야 한다는 결론을 얻을 수 있었다.
   2. **향후 연구 방향**
      1. 실험 계획  
         : **[결론 i]**에서와 같이 concat의 입력 텐서 그룹수와 보존율과의 연관성을 고려할때 Bottleneck의 반복 수 증가에 의한 보존율 감소 효과를 동시에 고려해야 한다. 그리고 향후 실험할 모델에서 Backbone의 Bottleneck 모듈 반복수를 변경할 때 과도한 반복 수 증가는 성능의 한계점과 별개로 양자화 과정에서 오차를 많이 유발해 NPU에서의 성능을 낮출 수 있다는 것 또한 염두하여 분석할 것이다.
      2. 필요 연구  
         : Bottleneck 반복 수와 보존율과의 상관 관계는 파악했지만 정확한 인과 관계에 대해서는 추가적인 분석이 필요하다.
2. **참조 문헌**

[1] 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.