**Backbone 구조 변경 분석 보고서**

1. **서론**
   1. 배경
      1. BackBone의 역할 및 yolo 구조 분석  
         : YOLO 모델은 크게 Backbone, Neck, Head 세 부분으로 이루어져 있다. Backbone은 이미지에서 특징 추출을 담당하는 부분이다. 이때, Backbone에서 C3/C2f 모듈 내에 있는 Bottleneck 모듈은 Feature에 대해 더 많은 정보를 추출하는 역할을 한다. 본 분석 보고서는 Backbone에서의 Bottleneck 모듈 반복 횟수를 조정할 수 있는 ‘n’을 변경하여 Bottleneck의 반복 횟수에 따른 성능 변화를 분석한다.
   2. 실험 준비
      1. 실험 항목
         1. v5n\_1111\_ta  
            v5n\_1221\_ta

v5n\_2222\_ta

v5n\_2332\_ta

* + - 1. v8n\_1111\_ta

v8n\_1221\_ta

v8n\_2222\_ta

v8n\_2332\_ta

* 1. 목적
     1. 백본에서의 Bottleneck 모듈 반복횟수 조정 시 성능 변화 분석  
        - 가설 : 백본에서 사용되는 Bottleneck 모듈의 반복 수를 늘리면 성능이 증가할 것이다.  
        : 실험 항목 1 => yolov5n 모델에 대해 반복 횟수 조정  
         실험 항목 2 => yolov8n 모델에 대해 반복 횟수 조정  
        위 실험 항목의 비교 기준으로 Bottleneck 반복 횟수 조정에 따른 객체 크기별 성능 변화 분석
  2. 평가 지표
     1. NPU 성능, GPU 성능

1. **본론**
   1. 실험 설계
      1. 실험 방법
         1. 동일한 데이터로 학습한 모델에 대해 동일한 입력 데이터를 사용하여 실험 모델 별 성능 측정.
            1. 학습 데이터 : 제공받은 train set과 add data set을 병합

train set : 기본으로 제공받은 데이터 셋

add data set : train set의 클래스에 맞춰 image-label쌍을 직접 수집한 추가 데이터 셋

* + - * 1. 입력 데이터 : 제공받은 test est
      1. 실험할 주제 외의 항목은 통제.
  1. 실험 결과 및 분석
     1. 실험 1 v5모델 Backbone의 Bottleneck 반복 횟수에 따른 성능  
        1. 비교 모델 : v5n\_1111\_ta vs v5n\_1221\_ta vs v5n\_2222\_ta vs v5n\_2332\_ta
        2. 모델 설명 :
           1. v5n\_1111\_ta : yolov5n 모델의 Backbone에서 C3 모듈 내의 Bottleneck 반복 횟수를 (1, 1, 1, 1)번 반복으로 변경한 모델
           2. v5n\_1221\_ta : yolov5n 모델의 Backbone에서 C3 모듈 내의 Bottleneck 반복 횟수를 (1, 2, 2, 1)번 반복으로 변경한 모델
           3. v5n\_2222\_ta : yolov5n 모델의 Backbone에서 C3 모듈 내의 Bottleneck 반복 횟수를 (2, 2, 2, 2)번 반복으로 변경한 모델
           4. v5n\_2332\_ta : yolov5n 모델의 Backbone에서 C3 모듈 내의 Bottleneck 반복 횟수를 (2, 3, 3, 2)번 반복으로 변경한 모델
        3. 가설   
           : Bottleneck 반복 횟수가 증가할수록 성능이 증가할 것이다.  
           Residual 형식인 Bottleneck 구조는 Gradient Vanishing 문제를 방지하고, 학습을 안정시키는 역할을 한다. 이러한 Bottleneck의 반복 횟수를 증가시키면 학습량이 증가하지만 이미지에서의 더 깊은 특징을 학습하여 많은 정보를 얻을 가능성이 있다고 본다.  
           따라서 Bottleneck 반복 횟수를 증가시키면 성능 또한 증가할 것이다.
        4. 결과 :   
           1. v5n\_1111\_ta

GPU 성능 : 55.40

NPU 성능 : 27.04

* + - * 1. v5n\_1221\_ta

GPU 성능 : 56.46

NPU 성능 : 33.43

* + - * 1. v5n\_2222\_ta

GPU 성능 : 54.56

NPU 성능 : 31.00

* + - * 1. v5n\_2332\_ta

GPU 성능 : 56.14

NPU 성능 : 31.56

텍스트, 스크린샷, 그래프, 도표이(가) 표시된 사진

자동 생성된 설명

* + - 1. 분석 : 위 그래프는 모든 class에 대한 객체 사이즈별 정확도를 나타낸다. 실험 결과, 가설과는 다르게 모든 객체 사이즈에 대해 v5n\_1221 모델이 가장 높은 성능을 보였다. 이를 통해 Backbone의 Bottleneck 반복 횟수를 늘리면 더 깊은 특징을 추출할 수 있지만, YOLOv5n 모델에서 성능 개선의 한계점이 존재하기 때문에 일정 기준치 이상으로 모듈을 반복할 경우 성능이 개선되지 않을 수 있다는 인사이트를 도출했다.
    1. 실험 1 v8모델 Backbone의 Bottleneck 반복 횟수에 따른 성능  
       1. 비교 모델 : v8n\_1111\_ta vs v8n\_1221\_ta vs v8n\_2222\_ta vs v8n\_2332\_ta
       2. 모델 설명 :
          1. v8n\_1111\_ta : yolov8n 모델의 Backbone에서 C2f 모듈 내의 Bottleneck 반복 횟수를 (1, 1, 1, 1)번 반복으로 변경한 모델
          2. v8n\_1221\_ta : yolov8n 모델의 Backbone에서 C2f 모듈 내의 Bottleneck 반복 횟수를 (1, 2, 2, 1)번 반복으로 변경한 모델
          3. v8n\_2222\_ta : yolov8n 모델의 Backbone에서 C2f 모듈 내의 Bottleneck 반복 횟수를 (2, 2, 2, 2)번 반복으로 변경한 모델
          4. v8n\_2332\_ta : yolov8n 모델의 Backbone에서 C2f 모듈 내의 Bottleneck 반복 횟수를 (2, 3, 3, 2)번 반복으로 변경한 모델
       3. 가설   
          : Bottleneck 반복 횟수가 증가할수록 성능이 증가할 것이다.  
          Residual 형식인 Bottleneck 구조는 Gradient Vanishing 문제를 방지하고, 학습을 안정시키는 역할을 한다. 이러한 Bottleneck의 반복 횟수를 증가시키면 학습량이 증가하지만 이미지에서의 더 깊은 특징을 학습하여 많은 정보를 얻을 가능성이 있다고 본다.  
          따라서 Bottleneck 반복 횟수를 증가시키면 성능 또한 증가할 것이다.
       4. 결과 :   
          1. v8n\_1111\_ta

GPU 성능 : 56.40

NPU 성능 : 29.42

* + - * 1. v8n\_1221\_ta

GPU 성능 : 56.88

NPU 성능 : 33.20

* + - * 1. v8n\_2222\_ta

GPU 성능 : 56.58

NPU 성능 : 32.66

* + - * 1. v8n\_2332\_ta

GPU 성능 : 57.76

NPU 성능 : 34.42

텍스트, 스크린샷, 그래프, 도표이(가) 표시된 사진

자동 생성된 설명

* + - 1. 분석 : YOLOv8n 모델 또한 YOLOv5n 모델의 결과와 비슷한 경향을 보인다.  
         v5n과 v8n 모델의 실험을 통해 nano 모델은 경량화 모델이기 때문에 성능 개선에 한계점이 존재할 것이고 각 모델의 실험 항목에서 Bottleneck의 반복수는 (1, 2, 2, 1)이 가장 적합하다.

1. **결론**
   1. 주요 결론  
      사용할 모델과 scale에 따라 Bottleneck 반복 횟수 변화에 따른 성능 개선의 한계점이 존재할 것이다. Bottleneck의 반복 횟수를 결정하며 정확도와 속도의 trade off를 조정하는 ‘n’은 사용할 모델에 따라 다르게 설정하여 가장 적절한 값을 찾아야 한다.
   2. 최적 조합
      1. YOLOv5n : v5n\_1221
      2. YOLOv8n : v8n\_1221
   3. 향후 연구 방향  
      Neck에서의 Bottleneck 반복 횟수에 따른 성능 변화도 실험하여 모델 네트워크 전체에서 쓰이는 Bottleneck 반복 횟수를 최적의 조합으로 설계할 계획이다.
2. **참조 문헌 @@@@@@@@여기부터 작성@@@@@@@@**

[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