**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
         2. v5n\_1221
         3. v5n\_2222
         4. v5n\_2332
         5. v8n\_1111
         6. v8n\_1221
         7. v8n\_2222
         8. v8n\_2332
   3. **목적 : v5n과 v8n 모델에서의 Bottleneck 반복 횟수 조정에 따른 성능 변화 분석**
      1. 백본에서의 Bottleneck 모듈 반복횟수 조정 시 성능 변화 분석  
         : [실험1] ~ [실험4]를 비교하여 v5n 모델에서 Backbone의 Bottleneck 반복 횟수를 조정했을 때 성능의 변화를 관찰하고, [실험5] ~ [실험8]를 비교하여 v8n 모델에 대해서도 마찬가지의 방법으로 비교한다. 이 때 GPU에서의 성능, NPU에서의 성능 두가지 관점으로 분석한다.
   4. **평가 지표**
      1. Parameter 수
      2. NPU 성능, GPU 성능, NPU/GPU 비율
      3. FPS
      4. 객체 크기별 정확도
2. **본론**
   1. **실험 설계**
      1. 데이터셋
         1. 각 모델들이 학습에 사용된 데이터와 테스트에 사용된 데이터는 동일
            1. 학습 데이터 : 제공받은 train set과 추가로 add한 set을 병합
            2. 테스트 데이터 : 제공받은 test set
      2. 실험 설계

\* 1111 ~ 2332는 Backbone에서 C3/C2f 모듈 내의 Bottleneck 반복 Count

* + - 1. v5n\_1111 : Backbone에서의 Bottleneck 반복 Count - (1, 1, 1, 1)
      2. v5n\_1221 : Backbone에서의 Bottleneck 반복 Count - (1, 2, 2, 1)
      3. v5n\_2222 : Backbone에서의 Bottleneck 반복 Count - (2, 2, 2, 2)
      4. v5n\_2332 : Backbone에서의 Bottleneck 반복 Count - (2, 3, 3, 2)
      5. v8n\_1111 : Backbone에서의 Bottleneck 반복 Count - (1, 1, 1, 1)
      6. v8n\_1221 : Backbone에서의 Bottleneck 반복 Count - (1, 2, 2, 1)
      7. v8n\_2222 : Backbone에서의 Bottleneck 반복 Count - (2, 2, 2, 2)
      8. v8n\_2332 : Backbone에서의 Bottleneck 반복 Count - (2, 3, 3, 2)
  1. **실험 결과 및 분석**
     1. 실험 1 v5모델 Backbone의 Bottleneck 반복 횟수에 따른 성능
        1. 실험 항목
           1. 대조군1 : v5n\_1111 / 실험군1 : v5n\_1221
           2. 대조군2 : v5n\_1221 / 실험군2 : v5n\_2222
           3. 대조군2 : v5n\_2222 / 실험군2 : v5n\_2332
        2. **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.
        3. 결과

텍스트, 스크린샷, 도표, 다채로움이(가) 표시된 사진

자동 생성된 설명

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | v5n\_1111 | **v5n\_1221** | v5n\_2222 | v5n\_2332 |
| Parameter (M) | 2.41M | **2.46M** | 2.31M | 2.68M |
| GPU Map (%) | 55.40 | **56.46** | 54.56 | 56.14 |
| NPU mAP (%) | 27.04 | **33.43** | 31.00 | 31.56 |
| NPU/GPU (%) | 48.81 | **59.21** | 56.82 | 56.22 |
| FPS | 218.11 | **204.52** | 206.18 | 190.07 |
| Acc by Class (%) |  |  |  |  |

* + - 1. 분석  
         위 그래프는 Bottleneck 반복 횟수 차이에 따른 모델 별 GPU/NPU의 mAP 값을 나타낸다. 실험 결과, 가설과는 다르게 GPU/NPU의 mAP에 대해 v5n\_1221 모델이 가장 높은 성능을 보였다.  
         이를 통해, Backbone의 Bottleneck 반복 횟수를 늘리면 더 깊은 특징을 추출할 수는 있지만, YOLOv5n 모델에서 성능 개선의 한계점이 존재하기 때문에 일정 기준치 이상으로 Bottleneck 모듈을 반복할 경우 성능이 개선되지만은 않을 수 있다는 인사이트를 도출했다.
    1. 실험 1 v5모델 Backbone의 Bottleneck 반복 횟수에 따른 성능
       1. 실험 항목
          1. 대조군1 : v8n\_1111 / 실험군1 : v8n\_1221
          2. 대조군2 : v8n\_1221 / 실험군2 : v8n\_2222
          3. 대조군2 : v8n\_2222 / 실험군2 : v8n\_2332
       2. **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.
       3. 결과  
          텍스트, 스크린샷, 직사각형, 도표이(가) 표시된 사진

          자동 생성된 설명

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | v8n\_1111 | v8n\_1221 | v8n\_2222 | **v8n\_2332** |
| Parameter (M) | 2.90 | 3.01 | 3.02 | **3.44** |
| GPU Map (%) | 56.40 | 56.88 | 56.58 | **57.76** |
| NPU mAP (%) | 29.42 | 33.20 | 32.66 | **34.42** |
| NPU/GPU (%) | 52.16 | 58.37 | 57.72 | **59.59** |
| FPS | 181.74 | 165.44 | 162.03 | **146.01** |
| Acc by Class (%) |  |  |  |  |

* + - 1. 분석 : v5n 모델 실험 결과를 통해 도출한 “Bottleneck 모듈을 더 반복한다고 성능이 계속적으로 오르지 않을 것”이라는 분석과 다르게 v8n 모델은 Bottleneck 모듈의 가장 많은 반복 수를 가진 모델에서 좋은 성능을 보였다.  
         이를 통해, v8n 모델의 4가지 실험 항목은 v5n 모델과 다르게 Bottleneck의 반복 횟수를 늘렸을 때의 성능 개선 한계점에 도달하지 않았기 때문에 가장 많은 반복 수를 가진 모델의 성능이 높았을 수 있다는 인사이트를 도출했다.

1. **결론**
   1. **주요 결론**사용할 모델의 종류와 Scale에 따라 Bottleneck 반복 횟수 변화에 따른 성능 개선의 한계점이 존재할 것이다. Bottleneck의 반복 횟수를 결정하며 정확도와 속도 간의 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\_2332
   3. **향후 연구 방향**  
      YOLOv8s 모델 실험을 통해 가장 적합한 Bottleneck 반복 횟수를 탐색할 계획이다.  
      또한, Neck에서의 Bottleneck 반복 횟수에 따른 성능 변화도 실험하여 모델 네트워크 전체에서 쓰이는 Bottleneck 반복 횟수를 최적의 조합으로 설계할 계획이다.
2. **참조 문헌**

[1] H. Zhou, Y. Yu, K. Wang, and Y. Hu, "A YOLOv8-Based Approach for Real-Time Lithium-Ion Battery Electrode Defect Detection with High Accuracy," *Electronics*, vol. 13, no. 173, 2024. DOI: [10.3390/electronics13010173](https://doi.org/10.3390/electronics13010173).

[2] https://ropiens.tistory.com/44