YOLOv8s 모델 실험 분석 보고서

앞으로 여러 모델 네트워크 구조를 실험 및 분석하기에 앞서, 기본 모델부터 분석을 하는 것이 앞으로의 방향성을 정하는 데에 있어 기준이 된다고 생각한다. 따라서 우선적으로 YOLOv8s 모델에 대해 분석한다.

|  |  |  |
| --- | --- | --- |
| 모델 이름 | | v8s\_org\_train |
| 모델 사양 | | 186 layers  2.7M parameters  6.8 GFLOPs |
| mAP50 | GPU | 0.543 |
| NPU | 0.345 |
| FPS | | 151.5 |

1. 실험 개요
   1. 배경

Nextchip사로부터 받은 데이터셋으로 학습시킨 YOLOv8s 기본 모델로 evaluation을 해본 결과 버스, 트럭, 자전거, 오토바이에 대한 mAP 성능이 낮은 것을 확인했다. 또한 소형 객체에 대해 낮은 성능과 약 16ms 후반의 Inference time이 측정되는 것을 확인했다. 이러한 단점을 보완하고 NPU에서의 inference time을 개선하기 위한 2가지 접근법을 고안했다.

첫번째는 소형 객체에 대한 추가 데이터셋 수집을 통해 학습을 진행하여 소형 객체를 더욱 잘 탐지하는 목표로 하는 것이다.

두번째는 소형 객체에 대한 feature map extraction을 강화하는 방법, 즉 모델 네트워크의 소형 객체 탐지 부분을 강화하는 방법이다.

이 실험에서는 여러 논문들을 참고하여 얻은 Insight를 통해 160x160 Image에서 feature map을 추출하도록 P2 레이어를 추가로 쌓아 올려 소형 객체 특징을 더 추출하는 것을 목표로 하는 것이다.

- Reference

1. 모델 수정 전략 (가설 설정)
   1. v8s\_p2\_train

위의 ‘1. 실험 개요’의 ‘배경’에서 언급했다시피 기본 YOLOv8s 모델은 소형 객체 탐지에 낮은 성능을 보이기 때문에 성능 개선을 위한 2번째 접근법인 Feature map extraction 강화로 접근을 하면 모델 네트워크의 P3(80x80)부분에서 해결 방안을 찾아볼 수 있다. 이를 통해 Backbone의 P2(160x160)은 소형 객체 탐지에 유리한 Layer이기 때문에 이 부분의 Feature map과 Neck을 P2 (160x160)까지 Upsampling한 Feature map을 Concatenate 하면 소형 객체에 대해 더 많은 정보를 담을 것이라는 가설을 설정하여 레이어를 추가한다.

기존 모델의 P3 모듈에 비해 더 많은 소형 객체의 정보가 담기기 때문에 소형 객체 탐지에 유리할 것이라고 예상한다.

1. 실험 세팅
   1. 모델 구조

텍스트, 스크린샷, 도표, 번호이(가) 표시된 사진

자동 생성된 설명

* 1. 구조 설명

YOLOv8 모델의 Architecture를 보면 Backbone의 P3, P4, P5에서 Feature map을 추출하여 Neck에서 Concatenate을 한다. Backbone에서 추출한 Feature map은 다양한 크기와 해상도를 가지는데 Neck은 이러한 Feature map을 통합하고 보완하여 더 풍부한 정보를 생성하는 역할을 한다. 또한 다양한 크기의 객체를 탐지해야 하기 때문에 다중 해상도의 Feature map을 사용하여 소형 객체와 대형 객체 모두를 효과적으로 처리한다.

위 그림의 (b)를 보면, Backbone의 P2에서 추출한 Feature map과 Neck에서 추가한 P2(160x160) Layer를 Concatenate하여 바로 Head로 내보내지 않고, 다시 한번 Convolution Layer를 통해 80x80 Size로 Downsampling한다. 이후에 생성되어 있는 P3(80x80) Layer와 Concatenate 하여 가설대로 소형 객체에 대해 더 많은 정보를 담은 Layer를 Head로 내보낸다. 이외의 Layer는 기본 YOLOv8s 구조와 동일하다.

1. 실험 결과
   1. Model information

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 모델명 | Layer 수 | Parameters | GFLOPs | mAP |
| v8s\_org\_train | 168 | 11,127,906 | 28.4 | 0.629 |

텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

자동 생성된 설명

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 모델명 | Layer 수 | Parameters | GFLOPs | mAP |
| v8s\_p2\_train | 195 | 11,325,154 | 32.4 | 0.651 |

텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

자동 생성된 설명

* 1. 성능 측정
     1. mAP
        1. GPU

|  |  |  |  |
| --- | --- | --- | --- |
|  | v8s\_org\_train | v8s\_p2\_train | 성능 |
| GPU | 0.629 | 0.651 | 약 3.5% 증가 |
| NPU | 0.421 | 0.457 | 약 8.6% 증가 |
| 성능 | 약 49.4% 증가 | 약 42.5% 증가 |  |

* + 1. FPS 비교

|  |  |  |  |
| --- | --- | --- | --- |
|  | v8s\_org\_train | v8s\_p2\_train | 성능 |
| FPS | 59.10 | 50.21 | 약 15% 감소 |
| 7 |  |  |  |

1. 결과 및 비교
   1. 결과
      1. 전체, 클래스별, confusion matrix
   2. 비교 : v8s\_org\_100ep랑 비교하기
2. 결론 및 분석
   1. 분석
      1. 문제점
   2. 향후 연구 방향
   3. 결론: 무슨 모델인지, 성능 값, 문제점
3. 참고 문헌
   1. YOLOv8, GhostConv, CBAM 관련 논문 및 자료.
   2. APACHE5 NPU 기술 문서 및 관련 자료.
   3. 기타 경량화 관련 연구들