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.

* Model Information

|  |  |  |
| --- | --- | --- |
| 모델명 | | v8s\_org\_train |
| 모델 사양 | | 168 layers  11.1M parameters  28.4 GFLOPs |
| mAP50 | GPU | 0.629 |
| NPU | 0.421 |
| FPS | | 59.10 |

* 성능 측정
  + mAP

텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

자동 생성된 설명

* + FPS : 59.1 / Inference time : 16.92 ms
  + Confusion Matrix Normalized

스크린샷, 직사각형, 도표, 사각형이(가) 표시된 사진

자동 생성된 설명

1. 실험 개요
   1. 배경

Nextchip사로부터 받은 데이터셋으로 학습시킨 YOLOv8s 기본 모델로 evaluation을 해본 결과 car를 제외한 Class의 낮은 mAP 성능을 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**

Huangfu, Zhongmin, Shuqing Li\*, and Luoheng Yan. "[Ghost-YOLO v8: An Attention-Guided Enhanced Small Target Detection Algorithm for Floating Litter on Water Surfaces](https://www.techscience.com/cmc/v80n3/57888/html)". *Computer Modeling in Engineering & Sciences*, vol. 80, no. 3, 2024, pp. 3714-3728.

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

텍스트, 자동차, 스크린샷, 차량이(가) 표시된 사진

자동 생성된 설명스케치, 도표, 그림, 종이접기이(가) 표시된 사진

자동 생성된 설명

위 그림과 같이 Convolution 연산에서는 입력층과 가까운 Layer일수록 Input Image에서 이미지의 엣지나 경계선 등 기본적인 시각적 특징들(소형 객체)을 찾는다. 반면 출력층과 가까운 Layer일수록 Input Image에서 넓은 영역에서의 특징인 눈, 코, 의자, 나무, 고양이 등 일반화가 곤란한 구체적인 이미지의 표현을 찾는다.

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 | | v8s\_p2\_train |
| 모델 사양 | | 195 layers  11.3M parameters  32.4 GFLOPs |
| mAP50 | GPU | 0.651 |
| NPU | 0.457 |
| FPS | | 50.21 |

* 1. 성능 측정
     1. mAP

텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

자동 생성된 설명

* + 1. FPS : 50.21 / Inference time : 19.92 ms
    2. Confusion Matrix Normalized

스크린샷, 직사각형, 도표, 텍스트이(가) 표시된 사진

자동 생성된 설명

1. 결과 및 비교
   1. 결과
      1. v8s\_org\_train 클래스별 mAP50

all : 0.629

person : 0.633

car : 0.903

bus : 0.779

truck : 0.687

cycle : 0.187

motorcycle : 0.587-

* + 1. v8s\_p2\_train 클래스별 mAP50

all : 0.651

person : 0.661

car : 0.911

bus : 0.799

truck : 0.706

cycle : 0.217

motorcycle : 0.611

* 1. 비교 : v8s\_org\_train vs v8s\_p2\_train

|  |  |  |  |
| --- | --- | --- | --- |
|  | v8s\_org\_train | v8s\_p2\_train | 성능 |
| mAP (GPU) | 0.629 | 0.651 | 약 3.5% 증가 |
| mAP(NPU) | 0.421 | 0.457 | 약 8.6% 증가 |
| 복원률 | 약 66.9% | 약 70.2% | 약 0.3% 증가 |

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

1. 결론 및 분석
   1. 분석
      1. 문제점

YOLOv8s 기본 모델에 P2 Layer를 추가하여 P3에 concat한 모델을 사용한 결과, GPU 기준으로 mAP가 약 3.5%, NPU 기준으로는 약 8.6% 증가했다. 특히, P2 레이어를 추가한 모델은 NPU에서 mAP가 더 큰 성능 향상을 보였다. 그러나 GPU -> NPU 변환 시 성능 복원률을 비교해 보면, P2 Layer를 추가한 모델이 기본 모델모다 0.3%로 미세하게 증가했다. NPU에서 mAP 성능이 올라도 GPU -> NPU 변환 성능 복원률 또한 비슷한 폭으로 오르지 않는다는 문제를 발견했다. 또한 Real Time Object Detection에서 중요한 FPS가 약 15% 감소했다.

* 1. 결론

기본 모델 대비 P2 Layer를 추가한 모델이 NPU에서 성능이 높은 증가율을 보였다. GPU 대비 NPU 성능의 복원률에서는 0.3%의 증가로 미미했다. 이러한 실험 결과를 통해 P2 레이어 추가로 인한 연산량의 증가로 양자화 과정에서 생기는 연산 양자화 손실이 늘어 GPU 대비 NPU 성능의 복원률 개선 폭이 낮아졌을 것이라는 가능성을 시사한다.

* 1. 향후 연구 방향

‘P2 Layer는 추가하여 사용하되 Conv연산 대신 GhostConv를 사용하여 연산량을 감소시키면 연산 양자화 손실도 감소할 것이다.’ 라는 연구 방향이 도출됐다. 이 과정에서 연산량 증가와 연산 양자화 손실 증가에 대해 상관관계를 확실하게 할 수 없기 때문에 파생된 2가지 연구 방향을 설정한다.

* + - 1. 연산량을 감소시켜 모델 경량화
      2. 연산량 감소에 따른 연산 양자화 손실 감소폭의 상관관계 연구

1번 연구 방향은 연산량과 연산 양자화 손실을 독립적으로 생각하여 단순히 연산량만 다루며 모델을 경량화에 초점을 둔다.

2번 연구 방향은 연산량의 증감에 따른 연산 양자화 손실의 상관관계를 다루는 데에 초점을 둔다.