|  |
| --- |
| 넥스트칩(컬러01) |
|  |
| **AI 기술인재 선발대회 결과 보고서** |
|  |
| Date: 2024.12.01 |
|  |

**Team : 자비스**

내용

[1. Introduction 3](#_Toc183602586)

[1.1. Abstract 3](#_Toc183602587)

[1.2. Goal 3](#_Toc183602588)

[1.3. Constraints 3](#_Toc183602589)

[1.4. Abbreviation 3](#_Toc183602590)

[1.5. Reference 4](#_Toc183602591)

[2. Details 5](#_Toc183602592)

[2.1. Members 5](#_Toc183602593)

[2.2. Timline 5](#_Toc183602594)

[2.3. Environments 5](#_Toc183602595)

[3. Model 5](#_Toc183602596)

[3.1. Overview 6](#_Toc183602597)

[3.2. Data Analysis 6](#_Toc183602598)

[3.3. Architecture 6](#_Toc183602599)

[3.3.1. Backbone 6](#_Toc183602600)

[3.3.2. Neck 6](#_Toc183602601)

[3.3.3. Head 6](#_Toc183602602)

[3.4. Training 7](#_Toc183602603)

[3.5. Example(추가로 진행한 방법론이 있다면 작성) 7](#_Toc183602604)

[4. Experiments 7](#_Toc183602605)

[5. Conclusion 7](#_Toc183602606)

[6. Appendix 7](#_Toc183602607)

# Introduction

## Abstract

- 소형 객체 탐지 성능 향상 : YOLOv8의 기본 네트워크(v8s)에 P2 Layer를 추가하여 소형 객체에 대한 탐지 성능을 개선  
- 모델의 백본 구조 변경을 통한 성능 향상 : 백본의 Bottleneck 반복수를 변경  
- 경량화 : Neck에서 사용되는 모듈을 교체하여 성능 손실을 최소화 하면서 추론 속도를 향상

## Goal

- 모델 성능 향상  
: 기존 YOLO모델의 문제점인 소형 객체에 대한 탐지 성능 및 전반적인 모델 성능 향상

- 경량화  
: 향상된 모델 네트워크에서 성능 손실을 최소화하며 경량화를 진행

- 보존율 향상  
: NPU 환경에 친화적인 모델 네트워크로 수정하여 GPU에서의 성능을 NPU 환경에서도 최대한 유지하도록 구현

\* 보존율 : GPU mAP 성능 대비 NPU mAP 성능

- 양자화 기법 적용

: 제공받은 aiWare Studio Tool을 사용하여 다양한 PTQ 방식 적용을 통해 양자화 손실폭 최소화

## Constraints

- 데이터셋의 한계  
: 제공받은 데이터셋과 모델의 테스트가 NPU에서 진행되는 특성상 일반적인 시점의 데이터를 추가하여 학습을 진행한 경우 GPU에서는 성능이 향상되지만 NPU에서의 성능은 오히려 감소하는 것을 확인, 따라서 기존에 제공받은 데이터셋만 활용하여 모델의 학습을 진행  
  
- Detect단 구조 고정의 한계  
: YOLO 모델의 기본 아키텍처는 P3, P4, P5 단을 Detect단으로 내보내지만 소형 객체 탐지에 유리한 P2단의 Layer를 Detect단으로 내보내려는 시도 과정에서 H/W 제약 사항으로 인해 Detect 단의 아키텍처를 수정할 수 없었다.

- 양자화 적용의 한계

: 제공받은 플랫폼 aiWare Studio Tool 버전에서 선정 모델인 YOLOv8 모델에 대해 PTQ 양자화 기법을 적용할 수 없는 제약 사항이 존재했다. 따라서 모델 네트워크를 수정하여 양자화 과정에서 발생하는 데이터 손실을 최소화하는 방향으로 프로젝트를 진행.

## Reference

|  |  |  |
| --- | --- | --- |
| **문서번호(저자)** | **제목** | **출처** |
| |  | | --- | | Li et al. |  |  | | --- | |  | | |  | | --- | | A YOLOv8-Based Approach for Real-Time Lithium-Ion Battery Electrode Defect Detection with High Accuracy |  |  | | --- | |  | | https://github.com/ ultralytics/yolov5 |
| J. Fernandez-Marques et al. | Searching for Winograd-aware Quantized Networks | https://arxiv.org/abs/2002.10711 |
| B. Jacob et al. | Quantization and training of neural networks for efficient integer-arithmetic-only inference | https://arxiv.org/abs/1712.05877 |
| PyTorch Team | Quantization Accuracy Debugging Documentation | https://pytorch.org/docs/stable/quantization-accuracy-debugging.html |
| H. Zhou et al. | A YOLOv8-Based Approach for Real-Time Lithium-Ion Battery Electrode Defect Detection with High Accuracy | https://doi.org/10.3390/electronics13010173 |
| Ropiens Blog | YOLO and its Applications in Image Processing | https://ropiens.tistory.com/44 |

# Details

## Members

김인호 (팀장)  
: 일정 관리, 팀원 업무 수행 방향 지시, 네트워크 설계, 실험 계획 및 관리, 데이터 관리, 보고서 검수, 업무 자동화, 결과 분석  
  
김기현 (팀원)  
 : 모델 양자화 분석, 활성화 함수 분석, 프로젝트 플랫폼 분석  
  
김세찬 (부팀장)  
: 네트워크 설계, 기존 기술 리서치 및 분석, 실험 설계 및 세팅, 결과 분석  
  
이찬혁 (팀원)  
: 모델 변환 방법론 분석, 모델 양자화, 네트워크 분석 및 설계

## Timline

|  |  |
| --- | --- |
| Process | Timeline |
| YOLO모델 관련 리서치 | 2024.09.12 ~ 2024.09.26 |
| NPU 디바이스 활성화 | 2024.09.26 ~ 2024.10.3 |
| YOLO v5, v8 비교 분석 | 2024.10.4 ~ 2024.10.14 |
| 백본 P2 Layer 적용 실험 | 2024.10.15 ~ 2024.10.29 |
| 백본 구조 분석 및 재설계 | 2024.10.30 ~ 2024.11.3 |
| 모델 경량화 실험 | 2024.11.3 ~ 2024.11.9 |
| 모델의 보존율 향상 실험 | 2024.11.10 ~ 2024.11.26 |
| 양자화 방법론 연구 | 2024.11.10 ~ 2024.11.26 |
| 진행 내용 문서화 | 2024.11.27 ~ 2024.12.1 |

## Environments

OS: Windows 11

Platform: Colab

Data: 제공받은 nextchip dataset 사용

Classes Information: Table1 과 같음

Table Task's Classes

|  |  |
| --- | --- |
|  | Detect |
| Classes Number | 6 |
| Classes Details | person  car  bus  truck  cycle  motorcycle |
| Scene | Rearview |

# Model

## Overview

- 소형 객체 탐지 성능 향상을 위한 구조 변경  
: Neck의 구조를 변경하여 P3 단에 P2 Layer 정보를 통합한 아키텍처 설계  
  
- Backbone 구조 변경  
: P3단에 P2 Layer통합한 모델을 기반으로 Backbone의 Bottleneck 반복수를 조절해가며 성능을 향상  
  
- 모델 경량화  
: Neck에 사용되는 모듈을 GhostConv와 C3Ghost 모듈로 교체하여 Inference time 손실을 최소화  
  
- Quantization  
: aiWare Stuio Tool을 사용하여 양자화 과정에서의 오차율을 줄이기 위한 방법론으로 nnef와 stats에 접근하여 rescaling 시도, 이후 YOLOv5 anchor-free, anchor-based 두 방식에 대해 aiWare Studio에서 제공하는 PTQ method 적용 시도

## Data Analysis

**[1. 소형 객체 탐지 성능 향상 방안]** 폴더 하위에

**[1.0 데이터셋과 모델 성능 상관관계 분석 보고서]**에 서술

## Architecture

텍스트, 스크린샷, 도표, 평행이(가) 표시된 사진

자동 생성된 설명  
YOLOv8s 모델 아키텍처를 기반으로 Backbone, Neck을 수정하여 위 그림과 같이 모델 네트워크 설계했다. Backbone과 Neck의 아키텍처 수정 사항은 아래의 **[3.3.1. Backbone]**과 **[3.3.2. Neck]**에서 설명한다.

### Backbone

- Backbone 구조 변경

|  |  |
| --- | --- |
| [YOLOv8s] | [Jarvis Model] |
| 텍스트, 스크린샷, 폰트이(가) 표시된 사진  자동 생성된 설명 | 텍스트, 스크린샷, 폰트이(가) 표시된 사진  자동 생성된 설명 |

: Backbone의 Bottleneck 반복 횟수에 변화를 주어 소형 객체 탐지에 더 많은 정보를 추출하도록 아키텍처를 설계했다.

기존 YOLOv8 모델의 Backbone 구조를 살펴보면 C2f 모듈에 repeats 값이 (3, 6, 6, 3)으로 선언되어 있다. 선정된 YOLOv8s 모델은 Scale로 [depth: 0.33, width: 0.50, max\_channels: 1024]를 사용하는데 C2f 모듈 내의 Bottleneck 반복 횟수(n)를 결정하는 요인은 ‘depth’와 ‘repeats’다. 계산 과정은 다음과 같다.

|  |
| --- |
| **n(number) = round(depth \* repeats)** |

예를 들어, depth 값은 0.33으로 고정이고 repeats가 3이면 0.99가 되고 이 값을 반올림하면 1이 되므로 Bottleneck 반복 횟수는 1번이다. 기존 YOLOv8 모델의 Backbone에 대해 계산해보면 Bottleneck 반복 횟수는 (1, 2, 2, 1)번이 된다.

Jarvis Model은 P2, P3 단에서 더 많은 Feature를 추출하여 소형 객체 탐지에 대해 성능을 개선하고자 Backbone에서의 Bottleneck 반복 횟수를 (2, 2, 1, 1)번 반복하도록 수정했다.

**- 제출물 -  
[1.3 Backbone 구조 변경 분석 보고서]  
[1.4 P2 적용 및 Backbone 구조 변경 분석 보고서]**

**[2.1.1 Bottleneck 반복수와 보존율 분석 보고서]  
[2.1.2 Concat 텐서 그룹수와 보존율 분석 보고서]**

### Neck

- 소형 객체 탐지 성능 향상을 위한 구조 변경

|  |  |
| --- | --- |
| [YOLOv8s] | [Jarvis Model] |
|  |  |

: Neck 부분에 P2에 대한 Layer를 추가로 생성하여 Detect단으로 나가는 P3 단에 P2 Layer 정보를 통합하는 방식으로써 소형 객체에 대해 더 많은 정보를 추출하도록 아키텍처 설계했다.

기존 YOLOv8 모델의 Neck 구조를 살펴보면 P3, P2, P1에 대해 Feature를 가공하여 사용한다. 하지만 여러 논문에서 언급되어오듯 YOLO 모델은 소형 객체 탐지에 취약점이 존재한다. 때문에 소형 객체에 대한 Feature를 추출하여 사용할 필요가 있다.  
 160x160 픽셀을 다루는 P2 단은 소형 객체에 대한 Feature를 추출하는 역할을 한다. P2 Layer를 Detect 단으로 내보내는 작업은 H/W 제약 사항으로 인해 한계점이 존재하지만 P2의 정보를 활용하여 기존 P3 단에 Concatenation하면 Feature 정보가 추가되어 소형 객체 탐지 성능이 증가한다.

P2 Layer를 추가 생성하여 통합하는 과정에서 계산량 증가와 추론 속도 감소 현상이 발생했다. 이에 따라, Neck에서 경량 모듈인 GhostConv와 C3Ghost 모듈을 사용하여 모델을 경량화했다.

**- 제출물 -**

**[1.2 P2 기법 적용 분석 보고서]**

**[1.4 P2 적용 및 Backbone 구조 변경 분석 보고서]  
[2.2.1 GhostConv, Conv 분석 보고서]  
[2.2.2 C2f, C3, C3Ghost 분석 보고서]**

## Training

* 1. **학습 환경**
     1. 구글 코랩  
        - A100 사용
     2. 하이퍼 파라미터  
        - batch\_size : 32  
        - lr : 0.01  
        - Optimizer : SGD
  2. **모델 정보**
     1. Summary  
        - layers : 369  
        - parameters : 8.56M  
        - gradients : 8.56M  
        - GFLOPs : 25.9
     2. 모델 구조 (Yaml file)  
        **final\_submission/3. Model\_files/v8s\_P2\_2211\_gc-c3g.yaml** 참조
  3. **성능 평가**
     1. - GmAP50 : @@@ 일단 비워놓기 @@@  
        - NmAP50 : @@@ 일단 비워놓기 @@@

보존율 : @@@ 일단 비워놓기 @@@  
(NmAP50 / GmAP50) \* 100

# Experiments

1. 소형 객체 탐지 성능 향상 방안

1.0 데이터셋과 모델 성능 상관관계 분석

1.1 v5n, v8n 분석

1.2 P2 기법 적용 분석

1.3 Backbone 구조 변경 분석

1.4 P2 적용 및 Backbone 구조 변경 분석

2. 모듈별 양자화 보존율

2.1.1 Bottleneck 반복수와 보존율 분석

2.1.2 Concat 텐서 그룹수와 보존율 분석

2.2.1 GhostConv, Conv 분석

2.2.2 C2f, C3, C3Ghost 분석

\* 제출물 **[5. documents]** 하위 폴더 참조

# Conclusion

본 실험에서는 YOLO 모델을 기반으로 소형 객체 탐지 성능을 높이기 위해 P2 레이어 적용, Bottleneck 반복수 증가 기법을 사용하였다.

이로 인해 무거워진 모델은 Neck 부분의 모듈을 경량화하여 최적의 성능 조건을 충족했다.

\* 정량적 결과는 제출물 **[3. JARVIS MODEL 산출 보고서]** 및 **[exp\_list.csv]** 참조

1. **인과적 추론**

: P2 레이어 및 Bottleneck 반복수 증가로 인해 소형 객체의 세밀한 특성 학습이 가능해졌고, 이는 탐지 정확도의 주요 향상 요인으로 작용했다.

Neck 부분에서 경량화와 동시에 NPU 성능 보존을 통해 실행 속도를 향상시키고, 성능 효율성을 최대화하였다. GhostConv와 C3Ghost를 함께 사용한 구조는 기존 대비 계산량을 줄이면서도 모델의 정확도를 유지하는 데 효과적임을 확인했다.

전반적인 성능 개선으로 보았을 때 Backbone에서의 적절한 특징 학습과 Neck에서의 Layer 추가 및 경량화를 통해 큰 손실 없이 모델을 최적화했다.

1. **종합적 통찰**

: Bottleneck 반복 수를 조정하여 성능은 향상했으나 추론 속도는 감소했다. 이에 따른 Neck 경량화 작업을 통해 이를 상쇄시키며 모델을 설계했다. 따라서 정확도를 향상시키면서 추론 속도의 감소는 최소화 할 수 있었다.

GhostConv와 C3Ghost를 함께 사용하는 경량화 기법은 정확도의 감소를 최소화 하면서 모델의 실행 속도를 높이게 되었는데, 특히 NPU 환경에서 실행했을 때 그 효과가 더 크게 나타났다. 따라서 엣지 디바이스에서의 적합하도록 모델의 구조를 변경하는데 성공했다고 생각한다.