**JARVIS MODEL 산출 보고서**

1. **서론**
   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. Yaml  
         - Model\_files/v8s\_P2\_2211\_gc-c3g.yaml 참조
   3. **성능 평가**
      1. - GmAP50 : 65.71  
         - NmAP50 : 57.24
      2. 보존율 : 87.11%  
         (NmAP50 / GmAP50) \* 100
2. **본론**
   1. **모델링 배경(flow)**
      1. 연구 진행 과정
         1. 1.0 데이터셋과 모델 성능 상관관계 분석  
            : 데이터셋 클래스 비율 및 객체 크기별 성능 분석을 통해 YOLO 모델의 성능 한계를 진단하고 데이터 보강 및 구조 개선 방향을 제시
         2. 1.1 v5n, v8n 분석  
            : 첫 Conv 레이어의 커널 사이즈에 따른 성능 차이와 C2f, C3 모듈의 성능 분석
         3. 1.2 P2 기법 적용 분석   
            : P2 feature map 적용시 소형 객체 탐지 성능 변화 분석
         4. 1.3 Backbone 구조 변경 분석   
            : v5, v8 모델 Backbone의 Bottleneck 반복 횟수에 따른 성능 분석
         5. 1.4 P2 적용 및 Backbone 구조 변경 분석   
            : Backbone의 P2, P3 단 Bottleneck 반복 횟수에 따른 성능 분석
         6. 2.1.1 Bottleneck 반복수와 보존율 분석  
            : C3 내부의 Bottleneck 반복 수와 보존율 상관관계 분석
         7. 2.1.2 Concat 텐서 그룹수와 보존율 분석  
            : C2f 내부의 concat 입력 텐서 그룹수와 보존율 상관관계 분석
         8. 2.2.1 GhostConv, Conv 분석  
            : C2f와 C3, C3Ghost 모듈 적용시 Neck에서 Conv와 GhostConv의 성능 차이 및 경량화 효과 분석
         9. 2.2.2 C2f, C3, C3Ghost 분석  
            : Neck에서 Conv와 GhostConv 사용 시 C2f, C3, C3Ghost 모듈 변경에 따른 보존율 분석
   2. **활용 기법**
      1. P2 적용

|  |
| --- |
| 텍스트, 스크린샷, 도표, 번호이(가) 표시된 사진  자동 생성된 설명 |
| [P2가 적용된 구조] |

: 기본 v8s 모델과 P2를 적용한 모델을 비교하여 전체 성능 및 소형 객체 탐지 성능을 비교 분석한 후, P2를 적용하기로 결정

최종적으로 v8s 모델에 P2를 적용하여 v8s\_P2 모델 산출  
- 산출 과정은 **[1.2 P2 기법 적용 분석 보고서]**에 포함

|  |  |  |  |
| --- | --- | --- | --- |
|  | NPU mAP (%) | GPU mAP (%) | FPS |
| v8s\_P2 | 57.89 | 64.96 | 51.03 |

1. Backbone의 Bottleneck 반복 수 조정

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

: P2를 적용한 v8s\_P2 모델에서 Backbone의 Bottleneck 반복 수를 변경해가며 성능 차이 분석  
최종적으로 v8s\_P2 모델의 Backbone을 변경하여 v8s\_P2\_2221 모델 산출  
- 산출 과정은 **[1.4 P2 적용 및 Backbone 구조 변경 분석 보고서]**에 포함

|  |  |  |  |
| --- | --- | --- | --- |
|  | NPU mAP (%) | GPU mAP (%) | FPS |
| v8s\_P2\_2221 | 57.96 | 66.05 | 49.69 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | NPU mAP (%) | GPU mAP (%) | FPS |
| v8s\_P2\_2211\_gc-c3g | 57.24 | 65.71 | 54.67 |

1. 경량화  
   : v8s\_P2\_2221 모델의 P3단 Bottleneck 반복 수를 변경하고, Neck에 GhostConv와 C3Ghost를 적용하여 경량화 구현.  
   최종 모델인 v8s\_P2\_2211\_gc-c3g\_140epoch 모델을 산출 (150epochs을 학습시켜 140epoch의 가중치를 추출)
2. **결론**
   * 1. P2 적용 효과  
        : P2 feature map을 적용한 v8s\_P2 모델이 기본 v8s 모델 대비 소형 객체 탐지 성능 및 전체 성능 개선에 효과적임을 확인했다.
     2. Backbone의 Bottleneck 반복 수 변경 효과  
        : Backbone의 Bottleneck 반복 수를 변경한 v8s\_P2\_2221 모델이 v8s\_P2 모델 대비 성능이 개선된 것으로 보아 P2 단의 적절한 Bottleneck 반복 수 증가가 성능 향상에 효과적이라고 판단했다.  
         하지만 성능 향상에 따라 FPS 값이 감소하여 하한선을 넘기게 되었다. 따라서 해당 모델을 경량화 시켜야 된다는 결론을 얻었다.
     3. 경량화 효과  
        : P3 단의 Bottleneck 반복 수를 줄이고 Neck에 GhostConv와 C3Ghost를 적용하여 최종 모델인 v8s\_P2\_2211\_gc-c3g 모델을 산출하였다.  
         NPU mAP는 0.9% 정도로 소폭 감소하였으며 FPS 값이 증가하여 경량화와 성능의 균형을 확보하였다.
3. **고찰**
   1. **분석적 사고**: 프로젝트를 진행하는 과정에서 그 동안은 커스텀 데이터로 학습하는 정도로만 사용해본 YOLO 모델에 대해 내부에 선언된 클래스들과 메소드들을 필요한 부분을 찾아 하나하나 분석해가며 기존에 배웠던 지식들이 현업에 어떻게 활용되고 있는지를 알 수 있었다. 또한, 모델 성능 향상 및 최적화라는 단순한 과제에 얼마나 많은 노력이 들어가야 하는지 알게 되었다.  
       우리의 생각이 맞다는 것이 입증되기도 하고 때론 반대의 결과가 나오기도 했지만 그 결과들 마주하며 언제나 ‘Why’와 ‘How’를 고민하며 공학도로서 필요한 분석적 사고에 한층 다가갈 수 있는 경험이었다.
   2. **엔지니어 마인드셋**: 기존의 생각이 맞을 때도 있지만, 반대 혹은 해석할 수 없는 결과가 나오는 경우도 너무 많았다. 하지만 편한 길을 찾지 않고 그 원인이 무엇일지 파고 들고 다시 추가로 필요한 실험들을 계획하며 포기하지 않고 성과를 내는 과정에서 좌절보다는 할 일이 늘어났을 때의 설레는 감정을 느낄 수 있었다. 향후 엔지니어로서 성장하는데 중요한 밑거름이 되는 두 달을 보냈다고 느꼈다.
   3. **아쉬운점**: 소형 객체 탐지 성능 향상과 경량화라는 목적을 달성하기 위해 여러 시도를 하였다. P2 feature map을 detect단 쪽에 여러 번 더해주는 방법, 기존의 Neck 구조를 반전시켜 적용한 구조, Upsample을 사용하지 않거나, 횟수를 줄이는 방법 등 여러가지 방법을 적용하였으나 결국 FPS 값이 낮거나, 성능이 낮은 경우가 대부분이었다. 결과적으로는 실패한 시도들이었지만 가설을 세우고 실험하며 결과에 대한 원인을 규명하는 과정에서 많은 것을 배웠다고 생각한다.