**GhostConv, Conv 분석 보고서**

1. **서론**
   1. **배경**
      1. 경량화  
         : 모델 네트워크 구조를 수정할 때, 성능 향상을 위해 모델을 변경하는 과정에서 FPS 하한선을 넘겨 real time & on-device에서 모델이 제 역할을 하지 못하게 된다. 따라서 모델 네트워크를 수정할 때는 모델을 경량화 하는 것이 중요하다.  
          특징 추출을 하는 가장 기본적인 연산인 Conv에서 경량화를 목적으로 파생된 GhostConv를 적용해 경량화 정도를 분석하여 모델 설계 시 본 보고서의 내용을 참조할 수 있도록 한다.
      2. Conv와 GhostConv의 구조적 차이  
         : Conv와 GhostConv는 feature map을 생성하는 방식에서 구조적 차이가 있다.  
         Conv는 입력 데이터를 모든 필터로 직접 연산해 feature map을 생성하는 방식이다.  
          그런데 GhostConv는 연산량을 줄이기 위해 일부 필터로 주요 특징만 추출한 뒤, 추가적인 특징은 기존 feature map을 활용해 생성한다. 그 후, 기존 feature map을 기반으로 생성된 추가 특징들을 결합하여 최종 출력 feature map을 생성한다.  
          따라서 GhostConv는 Conv에 비해 연산량을 줄일 수 있어, 경량화에 유리한 구조를 가지기 때문에 본 실험은 해당 Layer를 사용해 경량화를 진행한 후 모델의 성능 변화를 분석한다.
   2. **실험 준비**
      1. 실험 모델
         1. v8s\_P2\_N c c2f
         2. v8s\_P2\_N c c3\_tv
         3. v8s\_P2\_N c c3g\_tv
         4. v8s\_P2\_N gc c2f\_tv
         5. v8s\_P2\_N gc c3\_tv
         6. v8s\_P2\_N gc c3g\_tv
   3. **목적 : Conv와 GhostConv의 성능 차이 분석**
      1. 실험 1 : C2f 적용 시 Conv와 GhostConv의 성능 분석  
         : C2f를 사용하는 [모델 1]을 기준으로 Conv을 GhostConv로 변경한 [모델 4]을 비교한다. 결과를 통해 C2f를 사용했을 때 GhostConv와 Conv의 성능 차이를 분석한다.
      2. 실험 2 : C3, C3Ghost 적용 시 Conv와 GhostConv의 성능 분석  
         : C3, C3Ghost를 사용하는 [모델 2],[모델 3]을 기준으로 Conv을 GhostConv로 변경한 [모델 5],[모델6]을 비교한다. 이때 Feature map을 생성 및 추출하는 Backbone의 구조는 모두 실험 1과 같이 C2f, Conv로 통일하고 feature map을 조합하는 Neck에서만 Conv에서 GhostConv로 변경한다. 결과를 통해 C3, C3Ghost를 사용했을 때 GhostConv와 Conv의 성능 차이를 분석한다.
   4. **평가 지표**
      1. 모델 성능 : NPU mAP, FPS
      2. 보존율( = (NPU mAP / GPU mAP) \* 100, GPU 성능 대비 NPU에서의 재현율)
2. **본론**
   1. **모델 설계**
      1. 데이터셋
         1. 각 모델들이 학습에 사용된 데이터와 테스트에 사용된 데이터는 동일
            1. 학습 데이터 : 제공받은 train set과 validation set을 병합
            2. 입력 데이터 : 제공받은 test set
      2. 실험 설계

\* sP2는 s 모델에 P2를 적용

* + - 1. v8s\_P2 : Neck에 Conv, C2f를 적용한 모델
      2. v8s\_P2\_N c c3 : Neck에 Conv, C3를 적용한 모델
      3. v8s\_P2\_N c c3g : Neck에 Conv, C3Ghost를 적용한 모델
      4. v8s\_P2\_N gc c2f : Neck에 GhostConv, C2f를 적용한 모델
      5. v8s\_P2\_N gc c3 : Neck에 GhostConv, C3를 적용한 모델
      6. v8s\_P2\_N gc c3g : Neck에 GhostConv, C3Ghost를 적용한 모델
  1. **실험 결과 및 분석**
     1. 실험 1 : C2f 적용 시 GhostConv와 Conv의 성능 차이 분석
        1. 실험 항목
           1. 대조군1 : v8s\_P2 / 실험군1 : v8s\_P2 \_N gc c2f
        2. **가설 – C2f 적용 시 GhostConv를 사용했을 때 모델이 더 가벼워진다.**

: **[배경ii]**에서 언급했듯이 GhostConv는 Conv보다 작은 단위로 연산을 수행하여 연산량이 적기 때문에 GhostConv를 적용한 모델이 가벼워질 것이다. NPU mAP의 경우 경량화가 되기 때문에 GhostConv을 사용했을 때 성능이 더 낮을 것이다.

* + - 1. 결과 :

|  |  |  |
| --- | --- | --- |
|  | c c2f | **gc c2f** |
| 보존율 (%) | 71.28 | **69.52** |
| NPU mAP (%) | 48.34 | **47.15** |
| FPS | 51.03 | **55.05** |

* + - * 1. 텍스트, 스크린샷, 도표, 라인이(가) 표시된 사진

           자동 생성된 설명 v8s\_P2 vs v8s\_P2\_N gc c2f

[그래프 1]

* + - 1. 분석   
         : **[그래프1]**를 보면 C2f를 적용한 경우, Conv를 사용했을 때가 GhostConv를 사용했을 때보다 성능이 높다.   
          GhostConv가 Conv 보다 연산량이 적어 경량화가 될 것이라 예상했지만 오히려 Conv를 사용했을 때의 FPS가 더 높다. 또한 NPU mAP와 보존율 또한 Conv를 사용했을 때가 GhostConv를 사용했을 때보다 성능이 좋다.   
          가설과 반대로 나온 원인으로는 C2f에서 특징을 추출하는 연산 방식이 GhostConv에서 특징을 처리하는 방식과 맞지 않기 때문인 것으로 보인다.
    1. 실험 2 : C3, C3Ghost 적용 시 GhostConv와 Conv의 성능 차이 분석
       1. 실험 항목
          1. 대조군1 : v8s\_P2 \_N c c3 / 실험군1 : v8s\_P2 \_N gc c3
          2. 대조군2 : v8s\_P2 \_N c c3g / 실험군2 : v8s\_P2 \_N gc c3g
       2. **가설 – C3, C3Ghost를 적용 시 GhostConv를 사용했을 때 모델이 더 가벼워진다.**

: **실험1**의 가설과 마찬가지로 GhostConv를 사용했을 때가 Conv를 사용했을 때보다 모델이 더 가벼울 것이다. **실험1**의 가설은 맞지 않은 것으로 결과가 나왔지만, C2f과 조합할 때 좋지 않은 것일 수 있기 때문에 여전히 가설은 위와 같다.

* + - 1. 결과

|  |  |  |
| --- | --- | --- |
|  | c c3 | **gc c3** |
| 보존율 (%) | 68.75 | **69.67** |
| NPU mAP (%) | 45.86 | **47.05** |
| FPS | 54.92 | **55.05** |

* + - * 1. 텍스트, 스크린샷, 도표, 라인이(가) 표시된 사진

           자동 생성된 설명 v8s\_P2\_N c c3 vs v8s\_P2\_N gc c3  
           [그래프 2]

|  |  |  |
| --- | --- | --- |
|  | c c3g | **gc c3g** |
| 보존율 (%) | 69.53 | **55.99** |
| NPU mAP (%) | 46.16 | **47.53** |
| FPS | 54.57 | **55.99** |

텍스트, 스크린샷, 도표, 그래프이(가) 표시된 사진

자동 생성된 설명B. v8s\_P2\_N c c3g vs v8s\_P2\_N gc c3g

[그래프 3]

* + - 1. 분석   
         : **[그래프2]**를 보면 C3를 적용한 경우, GhostConv를 사용했을 때가 Conv를 사용했을 때보다 성능이 높다. FPS에서는 오히려 차이가 작지만 NPU mAP에서 차이가 발생한다. 또한 **[그래프 3]**을 보면 C3Ghost를 적용한 경우, GhostConv를 사용했을 때가 Conv를 사용했을 때보다 성능이 높다.   
          두 실험에서 모두 GhostConv를 사용한 모델들이 경량화 되었을 뿐만 아니라 NPU mAP, 보존율 성능이 증가하였다. 특히 GhostConv와 Conv의 성능 차이가 C3보다 C3Ghost에서 차이가 더 크다.   
          **실험 1**과는 반대로 C3, C3Ghost 모두 GhostConv를 사용했을 때 모델이 가벼워지고 성능도 좋아진 것으로 보아 C3와 C3Ghost는 GhostConv와의 조합이 좋은 것을 시사한다. **실험 1**과 결과가 다른 이유로는, C3와 C3Ghost의 연산이 C2f 보다 결합되는 텐서 그룹 수가 적은 것이 GhostConv 연산에 적합한 것으로 보인다.  
          C3Ghost에서 특히 성능 차이가 발생하는 것의 원인으로는 C3Ghost는 C3, C2f보다 더 작은 단위로 연산을 나누어 진행하고, GhostConv 또한 Conv보다 작은 단위로 연산을 진행하는데, 작은 단위로 나누어 계산할수록 양자화 과정에서 발생하는 오차가 감소하기 때문이다.  
          따라서 GhostConv와 C3Ghost를 동시에 적용하는 것이 NPU 연산에 가장 친화적이라고 판단된다.

1. **결론**
   1. **주요 결론**
      1. 실험 1 : C2f 적용 시 GhostConv와 Conv의 성능 차이  
         : **실험 1** 에서 C2f 적용 시 Conv 연산과 더 적합하였다. C2f 연산 특성 상 Conv 연산과의 조합이 더 좋은 것으로 보였으며, C2f 기법을 사용하게 된다면 Conv 연산과의 조합이 모델 개선에 효과적이라는 결론을 얻었다.
      2. 실험 2 : C3, C3Ghost 적용시 GhostConv와 Conv의 성능 차이  
         : **실험 2** 에서 C3, C3Ghost 적용 시 GhostConv 연산과 더 적합하였다. 주 목적인 경량화도 되었으며 성능 또한 증가하였다. C3Ghost의 경우 특히 GhostConv와 같이 사용하면 C3를 사용했을 때보다 성능이 더 좋아지기 때문에 GhostConv를 사용할 시 C3Ghost와의 조합하는 것이 NPU mAP 향상에 효과적이다.  
         텍스트, 스크린샷, 다채로움, 도표이(가) 표시된 사진

         자동 생성된 설명
   2. **향후 연구 방향**
      1. 실험 계획
         1. 이후에 모델을 경량화 시키는 기법으로 GhostConv와 C3Ghost를 함께 적용하여 연구를 진행한다.
      2. 필요 연구
         1. C2f, C3, C3Ghost 구조와 Conv, GhostConv 구조에 대한 분석을 통해 C2f 적용 시 Conv와의 조합이, C3Ghost와 C3 적용 시 GhostConv와의 조합이 좋은 결과에 대한 정확한 원인에 대한 연구가 필요하다.