Vision Transformer를 활용한 분류 기반의 접근 방식을 통한 ReCAPTCHA v2 챌린지 해결

Solving ReCAPTCHA v2 Challenge using Classification-based Approach with Vision Transformer

김보미 (Bomi Kim)

한국외국어대학교 컴퓨터공학부 [kimbomi0819@naver.com](mailto:kimbomi0819@naver.com)

본 논문은 2023학년 6월에 제출된 한국외국어대학교 컴퓨터공학부 졸업논문입니다.

지도교수: 장익범 교수님

서명: 

Abstract

한글 요약 : 본 논문은 Vision Transformer(ViT)를 중심으로 한 분류 기반 접근 방식을 통해 Recaptcha 챌린지를 탐구합니다. YOLO를 활용한 기존 객체 탐지 기법의 한계를 극복하면서 뛰어난 성능 향상을 확인하였습니다. 연구 과정에서 ViT의 견고한 특성과 소형 객체 분류에서의 능력에 주목하였고, 이전 방법론들의 강점과 약점을 철저히 분석하였습니다. 실험 결과를 통해 본 연구의 방법론이 성능 향상을 도출하였으며, 소형 객체 분류와 관련된 기존 연구와의 명확한 차별성을 확인하였습니다. 이러한 결과는 ViT기반의 분류 전략이 Recaptcha 챌린지에 중요한 가치와 기여를 제공함을 보여줍니다.

핵심어 : Vision Transformer, 분류 기반 접근 방식, 소형 객체 분류, Recaptcha 챌린지

영문 요약 : This paper explores the ReCAPTCHA challenge through a classification-based approach centered on Vision Transformer (ViT). While overcoming the limitations of existing object detection techniques utilizing YOLO, notable performance enhancements have been confirmed. Throughout the research process, attention was paid to the robust features of ViT and its capabilities in classifying small objects, along with a thorough analysis of the strengths and weaknesses of previous methodologies. Experimental results demonstrate that the methodology of this study, based on ViT, has derived performance improvements and confirmed a clear distinction from existing research related to small object classification. These results illustrate that the ViT-based classification strategy provides significant value and contributions to the ReCAPTCHA challenge.

Keywords : Vision Transformer, classification-based approach, small object classification, Recaptcha challenge

**목차**

**1. 서론**

1.1 연구 배경

1.2 연구 주제 설명

1.3 기존 연구와 비교

1.4 Recaptcha challenge 유형 분류

**2. 연구 방법 및 결과**

2.1 데이터

2.2 Base model : Transfer learning based on Inception v3

2.3 모델 선정 : Vision Transformer

2.4 모델 훈련

2.5 Vision Transformer 테스트 결과

2.5.1 큰 이미지 분류 성능

2.5.2 작은 이미지 분류 성능 및 라벨링

2.6 YOLO 테스트 결과

**3. 결과 분석**

3.1 Vision Transformer과 YOLO 모델 성능 비교

3.2 YOLO의 문제점 및 해결 방안

3.3 실제 Recaptcha 시스템에 적용하여 테스트 (YOLO)

**4. 결론 및 향후 연구 방향**

4.1 결론

4.2 한계점 및 향후 연구 방향

1. 서론
   1. **연구 배경**

인터넷 사용량의 증가와 함께 보안과 관련된 문제가 점점 중요성을 더하고 있습니다. 특히, 자동화 봇에 의한 악용 사례가 증가하고 있으며, 이를 방지하기 위해 많은 웹사이트에서 ReCAPTCHA와 같은 보안 챌린지를 도입하고 있습니다.

스크린샷, 야외, 나무, 차량이(가) 표시된 사진

자동 생성된 설명

그림 1.1.1 : reCAPTCHA v2

ReCAPTCHA는 사용자가 인간인지 로봇인지를 판별하기 위한 시스템으로, 보통 이미지 내의 특정 객체를 인식하도록 설계되어 있습니다.

웹 보안의 핵심 부분인 ReCAPTCHA 챌린지는 이전 연구에서 주로 객체 검출 방식, 특히 YOLO에 의존하여 다루어졌습니다. 그러나, 이 기법은 bounding box의 정확한 지정과 관련된 다양한 문제점을 안고 있습니다. 예를 들어, 일부 초과 타일이 잘못 분류될 수 있으며, 이는 결국 잘못된 분류 및 증가된 계산 비용으로 이어집니다.

이 연구는 ReCAPTCHA 챌린지를 해결하기 위한 새로운 방법으로 Vision Transformer(ViT)를 활용한 분류 기반 접근 방식을 제시합니다. 이 접근 방식은 전체 이미지를 전역적으로 고려하고, 초과 타일과 관련된 특징을 더욱 정확하게 포착할 수 있습니다, 또한, 계산 비용을 절감하면서 다양한 객체 클래스를 효과적으로 예측하는 데에 유연성을 제공합니다.

* 1. **연구 주제 설명**

이 연구의 명확한 주제는 "ReCAPTCHA 챌린지를 객체 검출 방식이 아닌 분류를 통해 해결하는 접근 방식의 탐구"입니다. 기존의 객체 검출 방식 대신 분류 기반의 접근 방식을 사용하여 성능을 개선하고, ViT(Vision Transformer) 모델을 활용하여 ReCAPTCHA 챌린지의 해결을 탐구합니다.

이 연구는 “Vision Transformer(ViT)를 사용한 분류 기반의 ReCAPTCHA 챌린지 해결 전략”을 중심으로 합니다. ViT는 이미지를 작은 패치로 분리하고, 각 패치를 개별적으로 분류하여 ReCAPTCHA 이미지의 패턴을 이해합니다.

구체적으로, 연구는 다음과 같은 내용을 다룹니다:

1. ViT를 사용하여 ReCAPTCHA 이미지를 패치로 분리하고 각 패치를 개별적으로 분류합니다.
2. 이진 분류를 사용하여 ReCAPTCHA 챌린지를 해결합니다.
3. 제안된 전략을 기존의 YOLO 방식과 비교하여 성능을 평가합니다.

이를 통해, 분류 기반의 전략과 ViT의 가능성을 탐색하고, 기존 객체 검출 방식의 한계를 극복하며, 성능 개선의 새로운 방안을 제시합니다.

* 1. **기존 연구와 비교**

주제에 대한 이전 연구가 수행되었습니다. 세 가지 유형의 CAPTCHA를 기계 학습으로 해결하는 연구가 2016년에 수행되었습니다. 이 연구의 솔버는 70.8%의 성공률을 보였습니다. Google의 reCAPTCHA v2를 해결하기 위해 객체 감지 기반 솔버를 사용하는 또 다른 연구는 2020년에 수행되었으며 83.3%의 성공률을 보였습니다.

기존의 객체 검출 방식을 사용한 연구들은 ReCAPTCHA 챌린지를 해결하기 위해 주로 bounding box를 예측하는 방식을 채택하였습니다. 그러나 이러한 방식은 초과 타일로 잘못 분류되는 문제가 발생하고, 계산 비용이 높아지는 등의 한계점을 가지고 있습니다.

Ref\_[1][2]

스크린샷, 건물, 바퀴, 육상 차량이(가) 표시된 사진

자동 생성된 설명

그림 1.3.1 : bounding box가 너무 많은 그리드 셀을 포함하는 오토바이 이미지

본 연구는 분류 기반의 접근 방식을 사용하여 이러한 문제를 극복합니다. 이 방법은 ReCAPTCHA 이미지를 패치 단위로 분리하고, 각 패치를 독립적으로 분류하여 이진 분류 문제로 해결합니다. 이를 위해 Vision Transformer (ViT) 모델을 사용하여 각 패치의 특징을 추출하고 분류합니다.

논문의 연구 결과는 기존의 객체 검출 방식에 비해 이점을 제시합니다.

분류 기반의 접근 방식은 초과 타일로 잘못 분류되는 문제를 줄일 수 있습니다. 기존의 객체 검출 방식은 경계 상자에 포함된 그리드 셀의 수에 따라 초과 타일로 잘못 분류될 수 있지만, 분류를 통해 전역적인 이미지를 고려하면 초과 타일과 관련된 특징을 더 잘 파악할 수 있습니다.

이러한 결과는 Recaptcha 챌린지의 해결에 대한 새로운 시각을 제시하며, 작은 객체 분류에 ViT를 활용한 접근 방식의 유용성을 보여줍니다. 논문에서는 이러한 결과를 바탕으로 ViT를 적용한 분류 기반의 접근 방식을 제안하고, 이를 통해 Recaptcha 챌린지의 성능을 향상시키는 것을 목표로 설정하였습니다.

* 1. **Recaptcha challenge유형 분류**

이전 논문에서 4x4 크기보다, 3x3크기에서 더 많은 오류율을 낸 점을 비추어, 3x3크기로 분할된 이미지 챌린지에 초점을 맞추고 해보았습니다.

본 논문에서는 3x3 사이즈로 잘라진 분할된 이미지를 ‘작은 이미지’로, 원본의 전체 이미지를 ‘큰 이미지’로 표현합니다.

Google reCAPTCHA v2 이미지 챌린지는 두 가지 유형으로 구분할 수 있습니다.

첫번째 방법은 “**큰 이미지를 분류**”해내는 것입니다.

야외, 텍스트, 스크린샷, 차량이(가) 표시된 사진

자동 생성된 설명

그림 1.4.1 큰 이미지 분류 : ‘자전거’ 예측

해당 방법은 클래스 정보가 포함된 이미지를 모두 선택하여 분류하는 방식입니다. 이진 분류에서 자전거를 찾는 경우, 다른 클래스로 예측된 경우에도 해당 이미지가 자전거인지 아닌지를 구분하여 정확도를 높이는 방법을 시도하려고 합니다.

이러한 방식은 문제의 특성을 잘 이용하여 정확도를 향상시킬 수 있다는 장점이 있습니다. 예를 들어, 자전거를 찾아야 하는데, 다른 클래스로 예측된 경우에도 자전거인지 아닌지를 명확히 구분하여 예측 결과를 출력합니다. 다른 클래스로 예측되더라도 해당 이미지가 자전거인지 아닌지를 정확하게 구분합니다. 이를 통해 모델의 정확도를 더욱 높일 수 있습니다.

두번째 방법은 “사진 한 장을 분할하여 특정한 객체를 찾기”입니다. 저는 본 논문에서 이것을 “작은 이미지 분류”라고 칭하겠습니다.

야외, 하늘, 스크린샷, 텍스트이(가) 표시된 사진

자동 생성된 설명

그림 1.4.2 작은 이미지 분류

이 방법은 이미지의 일부 영역을 분할하여 특정한 객체를 찾는 것을 목표로 합니다.

1. 연구 방법 및 결과
   1. **데이터**

Kaggle (Ref\_[3])에서11,671개의 이미지를 수집하였습니다. 각 이미지는 Bicycle, Bridge, Bus, Car, Chimney, Crosswalk, Hydrant, Motorcycle, Other, Palm, Stair, Traffic Light과 같이 12개의 카테고리로 분류되어 있습니다.

* + Bicycle : 780 images
  + Bridge : 533 images
  + Bus : 1209 images
  + Car : 3496 images
  + Chimney : 124 images
  + Crosswalk : 1240 images
  + Hydrant : 955 images
  + Motorcycle : 81 images
  + Other : 1340 images
  + Palm : 911 images
  + Stair : 211 images
  + Traffic Light : 791 images

총 11,671개 이미지를 수집하였습니다.

텍스트, 야외, 스크린샷, 나무이(가) 표시된 사진

자동 생성된 설명

그림2.1.1 : 훈련데이터로, 각 class에 대한 크기가 224x224인 이미지 데이터 9개 표현

훈련데이터로는 12개의 class에 대해 9340개의 이미지를 사용하였으며, 검증 데이터로는 전체 데이터의 20%인 2331개의 이미지를 사용하여 분류 성능을 측정하였습니다.

‘Other’ 클래스는 훈련 데이터에서만 사용되며, 테스트 데이터에서는 해당 클래스의 분류를 수행하지 않았습니다.

첫번째 분류 방법인 ‘큰 이미지 분류’에서는 각 class당 600장의 이미지로 테스트 데이터를 만들었습니다. 예를 들어, bicycle을 고르라는 챌린지가 주어졌을 때의, bicycle사진 200장, bicycle이 아닌 사진 400장을 테스트 데이터로 설정하였습니다.

두번째 분류 방법인 ‘작은 이미지 분류’에서의 테스트 데이터는 각 class당 원본 이미지 중 900개를 3x3 크기로 잘라서 작은 이미지로 사용하여 이진 분류를 수행하였습니다. 그리고 테스트 데이터의 성능을 내기 위해 각 이미지를 각 class에서의 그림이 선택되는 것에 대해 이진 분류를 수행하기 위해 라벨링을 해주었습니다.

야외, 스크린샷, 텍스트, 바퀴이(가) 표시된 사진

자동 생성된 설명

그림 2.1.2: 테스트 데이터 : 작은 이미지 9개

패턴, 대칭, 다채로움, 예술이(가) 표시된 사진

자동 생성된 설명

그림 2.1.3 : 테스트 데이터 – 작은 이미지 : ‘Bicycle’클래스 900장

데이터 전처리 단계에서는 데이터 증강 기법을 사용하였습니다. 모든 이미지의 픽셀 값을 0에서 1로 정규화하고, 이미지 확대 및 축소 범위를 지정하여 다양한 시각적 변화를 도입하였습니다. 또한, 이미지의 좌우 대칭성을 고려하여 수평으로 뒤집기를 적용하였습니다.

* 1. **Base model : Transfer learning based on Inception v3**

ImageNet 데이터셋으로 사전 훈련된 가중치를 사용하여 Inception v3 모델을 기본 모델로 선택했습니다. 검증 결과 손실은 0.9744이고 정확도는 0.8130으로 베이스라인 성능을 설정했습니다.

* 1. **모델 선정 : Vision Transformer**

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

그림 2.3.1: Vision Transformer

Vision Transformer(ViT)는 이미지 분류를 위한 딥러닝 모델입니다. 이 모델은 기존의 CNN 대신 transformer 아키텍처를 이미지에 적용한 것이 특징입니다. 이미지는 고정된 크기의 패치로 자르고, 각 패치를 선형 임베딩하여 위치 정보와 함께 transformer 인코더에 입력합니다. 분류를 위해 classfication token을 추가되어 이미지의 전체적인 컨텍스트를 인코딩하고 클래스를 예측합니다.

ViT는 이미지의 전체적인 패턴과 변형을 학습하는 데 강점이 있어 reCAPTCHA v2 이미지를 분류하는 데 사용됩니다. ViT는 이러한 패턴을 상호작용을 통해 학습하고 분류할 수 있는 능력을 갖추고 있어서 선택하게 되었습니다. 패치 간의 관계를 모델링하면서 이미지의 전체적인 구조와 패턴을 이해하고 클래스를 예측하는 데 도움이 되기 때문입니다.

(Ref\_[4])

* 1. **모델 훈련**

저는 ViT 모델의 B\_32(base 32)구조를 선택했습니다. Vit\_b32는 상대적으로 작은 모델으로 알려져 있으며, 제한된 컴퓨터 리소스와 계산 효율성을 고려해 선택하였습니다. 또한, 대규모 데이터셋에서 사전 훈련된 가중치를 사용하여 성능을 향상시켰습니다. 입력 이미지는 32x32 크기의 패치로 분할되어 처리되었습니다.

실험 설정에서는 Adam 옵티마이저를 사용하였고, 학습률은 1e-4로 설정하였습니다. 또한 Early Stopping 기법을 적용하여 검증 손실이 3번 연속으로 개선되지 않을 경우 학습을 조기 종료하였습니다.

첫번째 방법인 ‘큰 이미지 분류’에서는 각 클래스 별로 분류를 수행하였습니다. 예를들어, ‘자전거’클래스의 경우, 자전거를 포함한 이미지를 1/3 비율로 샘플링하고, 다른 클래스에서 무작위로 선택한 이미지를 2/3비율로 샘플링하여 테스트에 사용했습니다. 이 과정에서 자전거가 아닌 이미지는 클래스 0으로 예측하고, 자전거가 포함된 이미지는 클래스1로 예측하게 됩니다.

두번째 방법인 ‘작은 이미지 분류’에서는 각 클래스 당 분류를 수행하였습니다. 테스트 데이터로는 작은 이미지를 각각의 class에 대해 900장을 테스트하였습니다. 즉, 각 클래스당 큰 이미지 100장으로 분류를 수행하였습니다.

* 1. **Vision Transformer 테스트 결과**

Recaptcha는 사용자에게 특정 class의 이미지를 선택하도록 요청하는데, 이 때의 상황을 고려하여 **recaptcha v2와 같은 시나리오에서의 성능 검증에 초점을 맞추어 test를 진행**하였습니다. 특정 클래스의 이미지에 대해, 이진 분류를 통해 모델의 분류 성능을 평가하여, 특정 클래스에 대한 모델의 민감도나 특정 클래스를 잘 걸러내는 능력을 보일 수 있습니다.

구체적인 방법으로는,

1. Test input dataset인 생성기 만들기
   1. 각각의 class에 대해 2개의 생성기를 만듭니다. 첫번째 생성기는 해당 클래스의 이미지만 포함합니다. 두번재 생성기는 해당 클래스를 제외한 나머지 모든 클래스의 이미지를 random하게 포함합니다.
2. Test 함수 짜기
   1. 예측값과 실제값을 통해 이진 분류로 해석합니다.

이렇게 이진 분류로 test하는 것은 매우 합리적으로 볼 수 있습니다.

1. 단순성 : 특정 이미지가 주어진 class에 속하는지 아닌지만 판별하면 되므로 복잡한 다중 클래스 분류보다 단순합니다.
2. 높은 정확도 : 이미지가 명확한 class에 속하지 않는 경우에는 이진 분류가 더욱 유리할 수 있습니다.
3. 효율성 : 각 이미지에 대한 분류 결정은 독립적으로 이루어져야하므로, 이진 분류 모델은 다중 클래스 분류 모델보다 계산적으로 더 효율적입니다.
4. 확장성 : 새로운 class를 도입할 때마다 새로운 이진 분류기를 학습시키기만 하면되므로 확장성을 고려했을 때 효율적입니다.

**2.5.1 큰 이미지 분류 성능**

검증 데이터의 정확도는 0.848, 손실은 0.576으로 나타났습니다.

테스트 데이터에서 각 클래스 별 정확도는 다음과 같습니다.

챌린지의 취지에 맞게, 큰 이미지 분류 테스트 결과를 보이겠습니다.

각 클래스 당, 그 클래스가 속한 것을 약 1/3으로, 그 클래스를 뺀 클래스들을 약 2/3으로 하여 이진 분류를 수행했을 때의 정확도, 조화평균값입니다. 예를 들면, ‘버스’를 고르라고 했을 때의 챌린지에 정답에 도달할 확률의 정확도, 조화평균값을 나타낸 것이라고 볼 수 있습니다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** | **Accuracy** | **F1-score about**  **‘해당 class’** | **F1-score about ‘해당 class 제외한 모든 class’** | **개수** |
| **Bicycle** | 0.99 | 0.98 | 0.99 | 600 |
| **Bridge** | 0.97 | 0.96 | 0.98 | 600 |
| **Bus** | 0.98 | 0.97 | 0.99 | 600 |
| **Car** | 0.95 | 0.93 | 0.96 | 600 |
| **Chimney** | 0.97 | 0.96 | 0.98 | 600 |
| **Crosswalk** | 0.95 | 0.92 | 0.97 | 600 |
| **Hydrant** | 1.00 | 1.00 | 1.00 | 600 |
| **Motorcycle** | 0.99 | 0.98 | 0.99 | 241 |
| **Palm** | 0.97 | 0.95 | 0.98 | 600 |
| **Stair** | 0.97 | 0.96 | 0.98 | 600 |
| **Traffic Light** | 0.95 | 0.93 | 0.97 | 600 |
| **Total** | **0.97** | **0.96** | **0.95** | **6051** |

표 2.5.1.1 : 각 클래스 별 큰 이미지 분류 성능

전체적으로, 큰 이미지 분류에서는97%의 정확도를 보였으며, 해당 클래스를 제외한 모든 클래스에 대한 f1-점수로는 95%로 높은 성능을 보였습니다. 또한 해당 클래스에 대한 f1-점수는 96%로 좋은 성능을 보였습니다.

전반적으로, 큰 이미지 분류에서는 좋은 성능을 보였으며, 각 클래스에 대한 분류 결과도 높은 정확도와 f1-점수를 보였습니다. 이는 모델이 다양한 클래스를 잘 구분할 수 있는 능력을 가지고 있음을 시사합니다.

그 중, Hydrant 클래스는 Accuracy와 F1-score 모두 1.00으로 완벽한 성능을 보입니다. 이는 모델이 소화전 이미지를 완벽하게 분류하고 있음을 의미합니다. 또한 Motorcycle 클래스는 다른 클래스들에 비해 적은 이미지 수(241개)를 가지고 있지만, 그럼에도 불구하고 높은 성능을 보여줍니다.

야외, 잔디, 텍스트, 거리이(가) 표시된 사진

자동 생성된 설명텍스트, 하늘, 야외, 건물이(가) 표시된 사진

자동 생성된 설명텍스트, 야외, 도로, 도로 표면이(가) 표시된 사진

자동 생성된 설명

그림 2.5.1.2 : 잘못 분류된 이미지

잘못 분류된 이미지들을 보면, 거의 두 물체가 같이있어 두 물체 중 하나로 다른 class를 예측한 것입니다. 이렇게 잘못 분류되는 것들을 보면 모델이 작은 객체까지 잘 인지하고 있다는 상태를 알 수 있습니다. 큰 이미지 분류에서는 이러한 이유로 성능이 감소하는 측면을 보여줄 수 있습니다. 그러나 **작은 이미지 분류에 사용할 때는 작은 객체까지 검출 가능하다는 측면에서 긍정적인 효과를 보일 수 있습니다.**

**2.5.2 작은 이미지 분류 성능 및 추가적 라벨링**

각 클래스당 분할된 이미지 900개, 즉 원본 이미지 100개에 대해 예측을 잘해내는지 보았습니다. 예외적으로, motorcycle은 데이터양이 부족하여 729개로 수행하였습니다.

이진 분류를 위해서 정답 레이블을 제가 라벨링해주었습니다.

텍스트, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명

다음으로는, 각 클래스별로 분류 성능과 어려움을 파악할 수 있는 분석 결과입니다.

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Accuracy** | **F1-score about**  **‘해당 class가 아닌 small image’** | **F1-score about**  **‘해당 class small image’** |
| **Bicycle** | 0.92 | 0.95 | 0.78 |
| **Bridge** | 0.80 | 0.86 | 0.67 |
| **Bus** | 0.80 | 0.87 | 0.54 |
| **Car** | 0.90 | 0.93 | 0.81 |
| **Chimney** | 0.92 | 0.95 | 0.72 |
| **Crosswalk** | 0.81 | 0.87 | 0.67 |
| **Hydrant** | 0.92 | 0.94 | 0.88 |
| **Motorcycle** | 0.87 | 0.92 | 0.58 |
| **Palm** | 0.88 | 0.91 | 0.81 |
| **Stair** | 0.78 | 0.87 | 0.29 |
| **Traffic Light** | 0.92 | 0.95 | 0.81 |
| **Total** | **0.87** | **0.91** | **0.68** |

표 2.5.2.1 : 각 클래스 별 작은 이미지 분류 성능

전체적으로, 약 9500개의 이미지에 대해, 작은 이미지 분류에서는 87%의 정확도를 보였으며, 해당 클래스가 아닌 작은 이미지에 대한 F1-점수는 91%, 해당 클래스인 작은 이미지에 대한 F1-점수는 68%로 평가되었습니다.

일부 클래스에 대해서는 작은 이미지 분류가 어려움을 겪는 것으로 나타났습니다. 예를 들어, Stair 클래스의 경우 정확도가 78%로 낮았고, 해당 클래스인 작은 이미지에 대한 f1-점수도 29%로 매우 낮았습니다. 이는 작은 이미지에서 계단을 정확하게 분류하기 어려웠음을 나타냅니다.

반면에, ‘Hydrant’클래스의 경우 정확도가 92%로 높았으며, 해당 클래스인 작은 이미지에 대한 f1-점수도 88%로 높은 성능을 보였습니다. 이는 작은 이미지에서 소화전의 분류를 아주 잘 수행된 것으로 보입니다.

* + Bicycle

Bicycle 클래스는 대체로 잘 분류되었습니다. 사람 머리와 자전거를 분류를 잘해내었으며, 작은 객체도 잘 잡아내었습니다. 하지만, 자전거가 모여있는 경우는 잘 분류하지 못하였습니다.

* + Bridge

굵은 다리를 잘 분류하는 경향이 있었습니다. 그러나, 어두운 이미지의 경우 인식이 어렵거나 매우 얇은 다리는 분류하지 못하는 경우가 있었습니다.

* + Bus

작은 객체는 잘 분류하지만, 확대된 이미지나 가려진 부분은 잘 분류하지 못하는 경향이 있었습니다.

* + Car

작은 객체를 잘 분류하며, 어두운 차도 잘 분류했습니다. 그러나 가려진 부분은 분류하지 못하는 경우가 꽤 있었습니다.

텍스트, 패브릭, 야외이(가) 표시된 사진

자동 생성된 설명야외, 텍스트, 잔디, 육상 차량이(가) 표시된 사진

자동 생성된 설명

그림 2.5.2.2: 차 - 작은 객체도 잘 분류해내는 car(1188), 가려진 것은 분류하지 못했습니다. (1206)

* + Chimney

건물, 벽돌, 벽, 야외이(가) 표시된 사진

자동 생성된 설명

그림 2.5.2.3 : 굴뚝 2개인 이미지

잘못 분류 : 큰 굴뚝을 잡아내지 못하고 작은 굴뚝을 잡아내었습니다.

* + Crosswalk

횡단보도의 검정색, 흰색인 대비되는 색깔을 잘 구분하여 분류하는 경향이 있었습니다. 그러나 블러 효과가 있는 이미지의 경우 분류가 어려웠습니다

횡단보도 외의 도로에 써있는 표시를 횡단보도로 잘못 인식하는 경향이 있었습니다.

* + Hydrant

잘 분류된 편이며, 작은 객체도 잘 인식하는 경향이 있었습니다.

* + Motorcycle

작은 객체를 잘 분류하는 경향이 있었습니다. 그러나 어두운 이미지에서는 분류가 거의 되지 않았습니다.

텍스트, 야외, 도로, 건물이(가) 표시된 사진

자동 생성된 설명

그림 2.5.2.4 : 오토바이 - 작은 객체 잘 분류

* + Palm

작은 객체를 잘 분리해내며, 야자수의 모양이 뚜렷한 점을 잘 인식하는 경향이 있었습니다. 또한, 블러 효과가 다른 클래스에 비해 객체를 잘 인식하는 편이었습니다.

야외, 육상 차량, 차량, 잔디이(가) 표시된 사진

자동 생성된 설명

그림 2.5.2.5 : 야자수 - 작은 객체 여러 개 잘 분류

식물, 나무, 야외, 야자과이(가) 표시된 사진

자동 생성된 설명

그림 2.5.2.6 : 야자수 - 많은 나무 중 야자수 잘 분류

야외, 하늘, 나무, 야자수이(가) 표시된 사진

자동 생성된 설명야외, 하늘, 식물, 야자과이(가) 표시된 사진

자동 생성된 설명

그림 2.5.2.7 : 야자수 - 가장자리 부분도 잘 분류

야외, 해변, 지상, 패브릭이(가) 표시된 사진

자동 생성된 설명

그림 2.5.2.8 : 야자수 - 블러 효과 다른 클래스에 비해 잘 캐치함

* + Stair

Stair 클래스의 분류는 어려웠습니다. 특히 그림자 진 계단 같은 경우에는 더 분류가 되지 않았습니다.

* + Traffic Light

불빛인 빨간색, 초록색을 잘 인식하며 분류가 잘 되었습니다. 어두운 이미지와 블러 효과가 있는 이미지에서도 비교적 분류가 잘 되었습니다. 비교적 불빛이 없는 신호등은 잘 인지를 못했으며, 표지판으로 인식이 잘못 될 때도 있었습니다. 하지만 작은 객체도 잘 인식하는 편이었습니다.

야외, 하늘, 나무, 식물이(가) 표시된 사진

자동 생성된 설명

그림 2.5.2.9 : 신호등 - 어두운 경우+ 블러 잘 분류, 작은 객체 잘 인지

야외, 텍스트, 하늘, 구름이(가) 표시된 사진

자동 생성된 설명

그림 2.5.2.10 : 신호등 - 잘못 분류 : 빨간색 신호등만 인식

Bicycle, Chimney, Hydrant, Traffic Light은 각각 명확하게 식별할 수 있는 특징을 갖고 있어 성능이 비교적 우수했습니다. 예를 들어, Hydrant는 특별한 색상과 형태를 가지고 있으며, Traffic Light는 세 개의 다른 색상의 불빛을 가져, 분류 작업에서 모델의 성능을 높일 수 있었습니다.

Stair, Bridge, Crosswalk는 다른 클래스와 형태나 특징이 비슷한 측며능로 보아, 성능이 낮았던 것으로 추측됩니다. 예를 들어, Stair는 다른 건물 또는 구조물의 일부로 인식될 수 있으며, Crosswalk는 도로나 다른 표면 패턴과 유사한 것으로 판단지어 질 수 있기 때문입니다.

Motorcycle은 샘플의 개수가 상대적으로 적기 때문에, 성능이 낮았던 것으로 추측됩니다.

위의 분석을 통해, 각 클래스 별로 분류 성능과 어려움을 파악할 수 있었습니다.

* 1. **YOLO 테스트 결과**

Vision Transformer와의 성능 비교를 보이기 위해, 똑같은 test방식을 사용하였습니다.

또한 이전 논문과 똑같이, yolo v5로 실험을 진행하였습니다. 가중치는 small보다는 large가 성능이 더 좋아서, large로 사용하였습니다.

YOLO v5는 COCO 데이터셋으로 사전훈련되어져있습니다. COCO 데이터셋의 클래스와 제 데이터셋과 일치하는 클래스는 6개로, 이것에 대해서만 테스트를 진행하였습니다.

큰 이미지 분류 성능을 test하는 것에 앞서, 각 큰 이미지들에 대해 객체 감지만의 성공률은 어떻게 되는지 먼저 실험해보았습니다.

|  |  |  |
| --- | --- | --- |
| Class | Detection Success Rate | Average Confidence Score |
| Bicycle | 61.15% | 0.63 |
| Bus | 34.49% | 0.66 |
| Car | 64.62% | 0.59 |
| Hydrant | 46.91% | 0.57 |
| Motorcycle | 58.79% | 0.64 |
| Traffic Light | 79.79% | 0.64 |
| Average Total | 57% | 0.62 |

어느정도 객체 인식을 하고는 있지만, 성능이 높진 않은 것으로 보입니다. 이것을 토대로 큰 이미지에 대해 recaptcha v2의 시나리오를 적용하여 테스트해보겠습니다.

* + 1. 큰 이미지 분류 성능 (YOLO)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | Accuracy | F1-score (Target class) | F1-score (Non-target class) | Count |
| Bicycle | 0.78 | 0.51 | 0.86 | 600 |
| Bus | 0.76 | 0.44 | 0.85 | 600 |
| Car | 0.76 | 0.51 | 0.84 | 600 |
| Hydrant | 0.92 | 0.86 | 0.94 | 600 |
| Motorcycle | 0.88 | 0.42 | 0.93 | 241 |
| Traffic Light | 0.81 | 0.66 | 0.87 | 600 |
| Average Total | 0.81 | 0.56 | 0.88 | 6051 |

추가적으로, 작은 이미지에 대한 YOLO 테스트는 성능 측정 면에서 하지 않았고, 이미지를 별도로 돌렸습니다. 그에 대한 결과는 밑의 2.7에 나와있습니다.

1. 결과 분석
   1. **Vision Transformer과 YOLO 모델 성능 비교**

Vision Transformer는 전체적으로 YOLO보다 더 뛰어난 성능을 보였습니다.

YOLO는 특히 타겟 클래스의 f1-score에서 떨어지는 성능을 보이며, 이는 객체 탐지와 분류 문제에 대한 YOLO의 접근 방식의 차이 때문이라고 추측됩니다.

1. Accuracy
   1. Vision Transformer : 평균 정확도는 약 0.97로 매우 높습니다. 모든 클래스에서 0.95 이상의 정확도를 보여줍니다.
   2. YOLO : 평균 정확도는 약 0.81로, ViT에 비해 낮습니다. 모든 클래스에서 정확도가 0.76~0.92 사이로, 비교적 불안정한 성능을 보입니다.
2. F1-score (Target Class)
   1. Vision Transformer : 0.96으로 매우 높습니다. 모든 클래스에서 0.93 이상의 F1-score를 달성합니다.
   2. YOLO : 0.56으로, ViT에 비해 상당히 낮습니다. 특히 Bicycle, Bus, Car, Motorcycle 클래스에서 0.42~0.51로 낮은 값을 보입니다.
3. F1-score (Non-Target Class)
   1. Vision Transformer : 0.98로 매우 높습니다.
   2. YOLO : 0.88로 ViT에 비해 낮지만, 그럼에도 불구하고 비교적 높은 성능을 보입니다.

눈에 띄는 class로 보자면, Hydrant와 Motorcycle입니다. Hydrant는 YOLO에서 상대적으로 높은 성능을 보입니다. ViT의 성능과 큰 차이가 없습니다. YOLO는 객체 검출 시, bounding box가 사각형으로 이루어져있으며 소화전의 모양이 사각형인 점이 더욱 검출 성능을 높인 것으로 보여집니다.

* 1. **YOLO의 문제점 해결 – Vision Transformer 예시**

YOLO의 문제점인, 초과타일 포함을 자세히 이미지로 보이겠습니다.

텍스트, 자동차, 육상 차량, 차량이(가) 표시된 사진

자동 생성된 설명

(1,3)(3,0) 초과타일

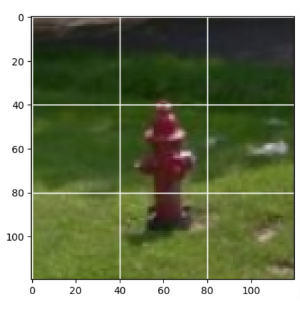
스크린샷, 텍스트, 사각형이(가) 표시된 사진

자동 생성된 설명

(0,1),(1,1) 초과타일

초과타일을 해결한, 몇 가지 예시를 이미지를 통해서 보여드리겠습니다.

1. Hydrant



원본 Hydrant이미지입니다. reCAPTCHA에서의 상황은 “소화전이 포함된 타일을 모두 선택하세요” 입니다.

* + - YOLO로 돌릴 시,

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* + - ViT로 돌릴 시,

텍스트, 소화전, 잔디, 스크린샷이(가) 표시된 사진

자동 생성된 설명

YOLO에서의 (0,1)을 bounding box에 의해 초과타일로 포함하여 클래스를 hydrant를 예측하게 됩니다. 하지만 ViT로 돌릴 시, (0,1)을 보면 이의 문제점을 해결해주는 것을 보일 수 있었습니다. 이는 이전 논문의 객체 탐지 문제점인 객체 타일 오버 탐지 부분에서 해결되었음을 보입니다.

1. Car

3x3, 4x4 에 대해 모두 보이겠습니다.

* + - ViT로 돌릴 시,



* + - YOLO로 돌릴 시,

스크린샷, 다채로움, 사각형이(가) 표시된 사진

자동 생성된 설명

YOLO의 3x3크기 이미지에서 (1,2)와, 4x4 크기 이미지에서 (0,2)(1,2)는 초과타일로 볼 수 있습니다. Vision Transformer는 이 초과타일을 not car로 잘 인지해내는 것을 보일 수 있었습니다.

1. 작은 객체 탐지

텍스트, 스크린샷, 하늘, 야외이(가) 표시된 사진

자동 생성된 설명

Vision Transformer의 경우 작은 객체를 어느정도 잘 잡아내는 것을 보였습니다. 하지만 YOLO로 돌릴 시, 아예 관계없는 class를 잘못 예측해내는 것을 보였습니다. YOLO는 ViT에 비해, 훨씬 작은 객체 인지에 있어서 성능이 떨어지는 것을 볼 수 있었습니다.

* 1. **실제 ReCAPTCHA 시스템에 적용하여 테스트 (YOLO)**

텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 소프트웨어, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

reCAPTCHA 이미지를 분석하여 정답 타일을 찾고, 인간처럼 보이는 웹 브라우저 상호작용을 통해 해당 타일을 클릭하도록 로컬에서 돌려보았습니다. 테스트 성능을 보였으니, YOLO에 대해서만 진행해보았습니다.

Google reCAPCHA demo 페이지에 접속한 후, selenium을 이용하여reCAPCHA 박스를 클릭하여 이미지 캡챠를 트리거하였습니다. 그 후, 캡챠 이미지를 스크린샷으로 저장하고, 저장된 이미지를 분석하여 정답 타일을 클릭하게 하였습니다.

Yolo의 detect.py를 변경해서 detect시, json 파일을 따로 생성하여 각 이미지당 예측된 label, confidence score, bounding box위치 정보를 포함하게 하였습니다. 그래서 이 bounding box의 위치 정보를 바탕으로, 어떤 grid에 속하고 있는지를 밝히고, 그 grid에 대해 정답으로 인식하게 하였습니다.

1. 결론 및 향후 연구 방향

**4.1 결론**

1. 성능 : Vision Transformer는 YOLO에 비해 전반적으로 더 높은 성능을 보였습니다. 특히, 타겟 클래스의 F1-score에서 큰 차이를 보였습니다. 이는 Vision Transformer가 이미지의 디테일한 정보를 더 잘 캡쳐하고, 분류 성능이 뛰어나다는 것을 나타냅니다.

1. YOLO의 특성 : 이전 논문에서 사용하였던, YOLO는 원래 실시간 객체 탐지를 목적으로 설계되었습니다. 즉, 여러 객체의 위치와 그 객체의 클래스를 동시에 예측하는 것에 최적화되어 있습니다. 반면, reCAPTCHA v2의 문제 설정에서는 단순히 특정 클래스의 객체가 이미지에 포함되어 있는지만 판단하면 됩니다. 이러한 task는 YOLO의 전체 기능이 필요하지 않을 수 있을 것으로 보입니다.
2. ViT의 특성 : ViT는 이미지를 여러 패치로 나누고 각 패치의 정보를 사용하여 이미지를 분류합니다. 이러한 방식은 전체 이미지에 걸쳐 발생하는 패턴과 관계를 더 잘 학습할 수 있기 때문에, reCAPTCHA v2와 같은 문제에서 효과적일 수 있습니다.
3. 효율성 : ViT는 대규모 데이터에서 훈련되었을 때 특히 더 높은 성능을 발휘합니다. 짧은 훈련시간을 거쳐 만들어진 저의 ViT 모델로도 비교적 높은 정확도와 f1-score을 보였습니다. 이는 ViT가 복잡하고 다양한 배경 속에서도 객체를 잘 분류할 수 있음을 나타냅니다.
4. 적용성 : reCAPTCHA v2와 같은 문제에는 단순한 이진 분류 방식이 더 적합할 수 있습니다. 여러 객체를 동시에 탐지하고 위치를 지정할 필요가 없기 때문입니다. 이러한 관점에서, ViT가 YOLO보다 더 적합한 선택이라고 볼 수 있습니다.

최종적으로, reCAPTCHA v2를 이진 분류 문제로보고, Vision Transformer를 사용하는 것이 더 적절한 선택과 성능을 보일 것으로 판단됩니다.

본 연구는 ViT 모델을 활용한 객체 검출에 대한 성능 평가와 ReCAPTCHA 챌린지 해결에 대해 분석하였습니다. ViT모델은 YOLO와 비교하여 객체 검출에서의 일부 단점을 보완하였으며, 분류 기반 접근 방식의 장점을 활용하여 ReCAPTCHA 챌린지 해결에 효과적인 도구로 활용될 수 있음을 확인하였습니다.

* 1. **한계점과 향후 연구 방향**

연구에서는 몇 가지 한계점을 발견하였습니다.

1. 라벨링의 어려움

연구 과정에서는 작은 이미지에 대한 라벨링 작업 시 사람이 봐도 구분하기 어려운 블러 효과나 너무 어두운 이미지와 같은 경우에 대해 애매한 상황이 발생하였습니다. 또한 ‘bridge’의 경우 다리의 정의를 어디까지로 설정해야 하는지에 대한 헷갈림이 있었습니다. 예를 들어, 다리에 연결된 기둥까지도 다리의 일부로 인식해야 하는지 여부에 대한 경계를 어디까지 설정해야 할지 애매했던 측면이 있습니다. 이는 주관적인 판단이 요구되는 부분으로, 라벨링 과정에서 개별 라벨러들 사이에 차이가 발생할 수 있음을 의미합니다.

이러한 어려움은 작은 객체의 경우에 특히 두드러지며, 불명확한 경계나 특징을 가진 객체의 경우에는 더 큰 어려움이 있을 수 있습니다. 따라서, 향후 연구에서는 이러한 어려운 경우에 대한 정확한 라벨링 가이드라인을 마련하고 라벨러들 간의 합의를 도모하여 객체 경계 설정의 일관성을 확보하는 것이 중요합니다. 또한, 이미지 품질 개선이나 블러 효과 보정 알고리즘 등을 적용하여 작은 객체에 대한 라벨링 정확도를 향상시킬 수 있는 연구도 필요합니다. 이와 함게, 라벨링 작업 시 참고할 수 있는 추가적인 도구나 시각적인 지침을 개발하여 명확한 기준에 따라 객체를 분류하고 경계를 설정할 수 있도록 지원하는 것도 고려해야 합니다. 이를 통해 작은 객체에 대한 라벨링 작업의 일관성과 효율성을 향상시킬 수 있을 것입니다.

1. Grid크기에 대한 추가적 연구 필요

ViT 모델은 patch size를 3으로만 적용한 3x3 그리드 챌린지에 대해서만 검증하였습니다. 4x4 그리드 챌린지에서의 성능에 대한 연구는 이전 논문에서 언급되었으나, 이번 연구에서는 해당 부분을 다루지 않았습니다. 따라서, 향후 연구에서는 4x4 그리드 챌린지에 대한 성능 분석을 수행하여 ViT 모델의 확장 가능성과 성능 변화를 확인하 것도 필요할 것으로 보입니다.

1. 자원의 한계

GPU의 제한된 자원으로 인해 모델 학습 및 추론에 소요되는 시간이 증가하여 어려움을 겪었습니다. 보다 강력한 GPU 자원을 확보한다면 더욱 발전된 모델을 만들 수 있을 것입니다.

위의 연구 방향을 통해 ViT 모델을 더욱 발전시키고, 객체 검출 및 ReCAPTCHA 챌린지 해결과 같은 다양한 응용 분야에서의 활용 가능성을 확장할 수 있을 것입니다.

References

[1] [Image Recognition for Solving Google’s reCAPTCHA An Investigation of how Different Aspects Affects the Security of Google’s reCAPTCHA](https://kth.diva-portal.org/smash/get/diva2:1703299/FULLTEXT01.pdf)  (Stockholm, Sweden 2022) [논문]

[2] [Object Detection based Solver for Google's Image reCAPTCHA v2](https://arxiv.org/pdf/2104.03366.pdf) (2021.4) [논문]

[3] Kaggle Dataset : <https://www.kaggle.com/datasets/mikhailma/test-dataset> [데이터셋]

[4] [AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE](https://arxiv.org/pdf/2010.11929v2.pdf) (ICLR, 2021) [논문]