**중고 거래 사기 피해를 방지하기 위한 합성 이미지 판별 시스템**

컴퓨터공학과 12181703 최희정

컴퓨터공학과 12181617 서민균

컴퓨터공학과 12184102 양유진

1. 배경

1.1 개발 동기

높아진 합성 기술력을 이용한 무분별한 합성으로 인하여 피해를 입는 사람들이 늘어나고 있다. 숙박업소의 사진 속 보이는 풍경이 합성된 사진인 경우도 있으며, 티켓을 합성하거나 없는 물건을 본인의 물건인 척 합성하는 사례들이 꽤나 있다. 이런 사례로 인한 소비자의 피해를 줄이기 위하여, 사진이 합성되었거나 조작되었는지 판별을 해주는 역할을 해주는 이미지 합성 판별기를 고안하였다.

1.2 개발의 필요성 및 기대효과

* 높아진 사진 합성 기술력을 악용한 범죄가 점점 많아지고 있다. 이를 통한 사기 피해를 방지하기 위한 서비스를 제공하고자 한다. 이 서비스의 이용자는 합성이 의심 가는 사진을 웹사이트에 업로드 하는 것만으로도 해당 사진이 합성인지 아닌지, 합성이라면 사진 중에 어떤 부분이 합성으로 의심되는지 확인할 수 있다.
* JPEG으로 압축된 이미지의 pixel 분포의 차이를 연구하여, 보다 높은 정확도로 합성된 이미지를 판별하고, 합성된 부분을 검출하는 학습 모델을 연구할 것이다.

2. 관련 연구 및 자료 조사

2.1 JPEG(Joint Photographic Experts Group) 압축

JPEG는 이미지 파일 형식의 종류 중 하나로 이미지를 색상 정보가 담긴 픽셀들로 표현하는 래스터(raster) 방식의 이미지 파일 형식이다. JPEG는 손실 압축 기술을 사용하는데, 손실 압축이란 원본 파일의 용량을 줄이기 위해 이미지를 고의적으로 손실시키는 것을 말한다. 여기서 중요한 것은 이미지의 화질을 최대한 유지하면서 손실 압축하는 것이다. JPEG의 압축 방식은 아래와 같은 순서로 이루어진다.

* 이미지 색 공간 변환

RGB 색공간을 YCbCr 색공간으로 변환한다.

* 다운 샘플링(Down-sampling) 및 서브 샘플링(Sub-sampling)

사람의 눈은 색상 정보보다 밝기 정보에 더 민감하기 때문에 밝기 정보(Y) 보다는 색상 정보(Cb, Cr)을 줄이는 방향으로 샘플링을 진행한다.

* 이산 코사인 변환(Discrete Cosine Transformation – DCT)

서브 샘플링 후에 각 채널을 8x8 블록으로 분할한 후 이산 코사인 변환(DCT)를 진행한다. 이 과정을 통해 주파수 관점에서 이미지를 볼 수 있게 된다. 주파수 도메인에서 가장 왼쪽 위의 값을 DC 계수라고 하고, 나머지 63개의 계수들을 AC 계수라고 한다. DC 계수는 저주파 성분과 관련되어 있으며, 해당 블록의 기본적인 명도를 결정하는 정보를 가지고 있다.

* 양자화(Quantization)

고주파 성분을 제거하기 위해 주파수 영역의 각 계수에 대해 특정 상수(양자화 행렬-quantization table)로 나눈 후 반올림한다. 전형적인 양자화 행렬(Standard quantization table)은 아래와 같지만, 많은 이미지 처리 소프트웨어들이 자신들만의 고유한 양자화 행렬(Non-standard quantization table)을 사용한다.

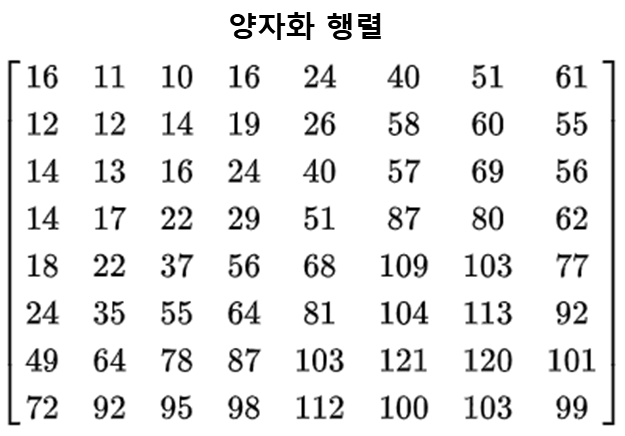


그림 1. Standard Quantization Table

* Zigzag 스캐닝

2차원으로 되어있는 양자화 된 DCT 계수들을 1차원의 데이터로 만든다.

* 양자화 된 DC/AC 계수 부호화

DC 계수들은 DPCM과 허프만 인코딩을 이용해 부호화 하며, AC 계수들은 런 렝스 부호화와 허프만 인코딩을 이용해 부호화 한다.

2.2 JPEG 이중 압축 탐지(Double JPEG compression Detection)

본 프로젝트에서는 합성된 사진의 통계적 분포를 분석한 후, JEPG 이중 압축 탐지 알고리즘을 활용하여 합성된 JEPG 사진에서 합성된 영역을 검출하는 것을 목표로 한다.

Double JPEG Detection in Mixed JPEG Quality Factors using Deep Convolutional Neural Network(Jin-Seok Park)의 논문에 따르면 정상적인 이미지 블록(block)들은 그림2의 Single JPEG block과 같이 원래의 양자화 행렬1(Q1)을 갖는 하나의 동일한 분포 특징만을 가진다. 하지만 이런 원본 이미지에 합성과 같은 조작이 이루어진 후 양자화 행렬2(Q2)로 재압축 되었다면, 조작된 영역은 Q2의 single JPEG 분포 특징을 가진다. 이와 반대로 조작되지 않은 영역은 Q1과 Q2로 두 번 압축된 특징을 가진다. 따라서 single JPEG 블록과 double JPEG 블록을 구분함으로써 의심스러운 영역을 찾을 수 있다. 이러한 방법으로 합성된 부분을 검출해내는 알고리즘을 JEPG 이중 압축 탐지 알고리즘이라고 한다.

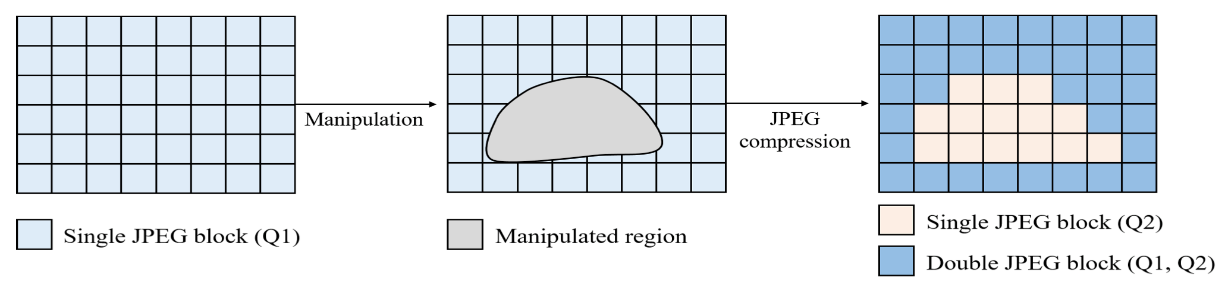


그림 2. 합성된 사진의 Double JPEG 압축 분포

이렇게 합성된 부분을 검출하기 위해서는 “single JPEG block”과 “double JPEG block”의 통계적 관점에서 차이가 있어야 한다. Double JPEG compression forensics based on a convolutional neural network(Qing Wang) 논문에서는 이러한 “single JPEG block”과 “double JPEG block”의 DCT 계수 histogram을 통하여 각각 둘에게서 다르게 나타나는 통계적 패턴 차이를 언급하고 있다.

각 주파수에서 single-compressed histogram은 대략적으로 일반적인 가우스 분포를 따르는 반면, double-compressed histogram은 가우스 분포를 따르지 않는다. 첫번째 압축에서의 quality factor를 Q1, 두번째 압축에서의 quality factor를 Q2라고 할 때, QF2 > Q1인 경우, 히스토그램은 주기적으로 값이 누락되어 있는 패턴을 보인다. 반면에, QF2 < QF1인 경우, 히스토그램은 주기적인 봉우리와 계곡 패턴(periodic peaks and valleys pattern)을 보인다. 이는 그림 3을 보면 확인할 수 있다.

텍스트, 보트이(가) 표시된 사진

자동 생성된 설명

그림 3. a, b: 단일 압축 이미지의 DCT 계수 histogram (a: QF1 = 60, b: QF1 = 90)  
c, d: 이중 압축 이미지의 DCT 계수 histogram (c: QF1 = 90, QF2 = 60, d: QF1 = 60, QF2 = 90

또한 해당 논문에서는 이미 DCT 상관계수로부터 추출해낸 histogram feature를 input으로 하고, 1D CNN을 활용한 네트워크를 사용할 때, 높은 정확도로 single JPEG 블록과 double JPEG 블록을 구분할 수 있다는 것을 실험적으로 입증하였다.

3. 개발의 목표 및 내용

3.1 개발하고자 하는 기술의 전체적 개념도

본 프로젝트에서 구현하고자 하는 시스템은 “합성 사진 판별 시스템”으로, 아래의 그림 4 와 같은 모습이다.

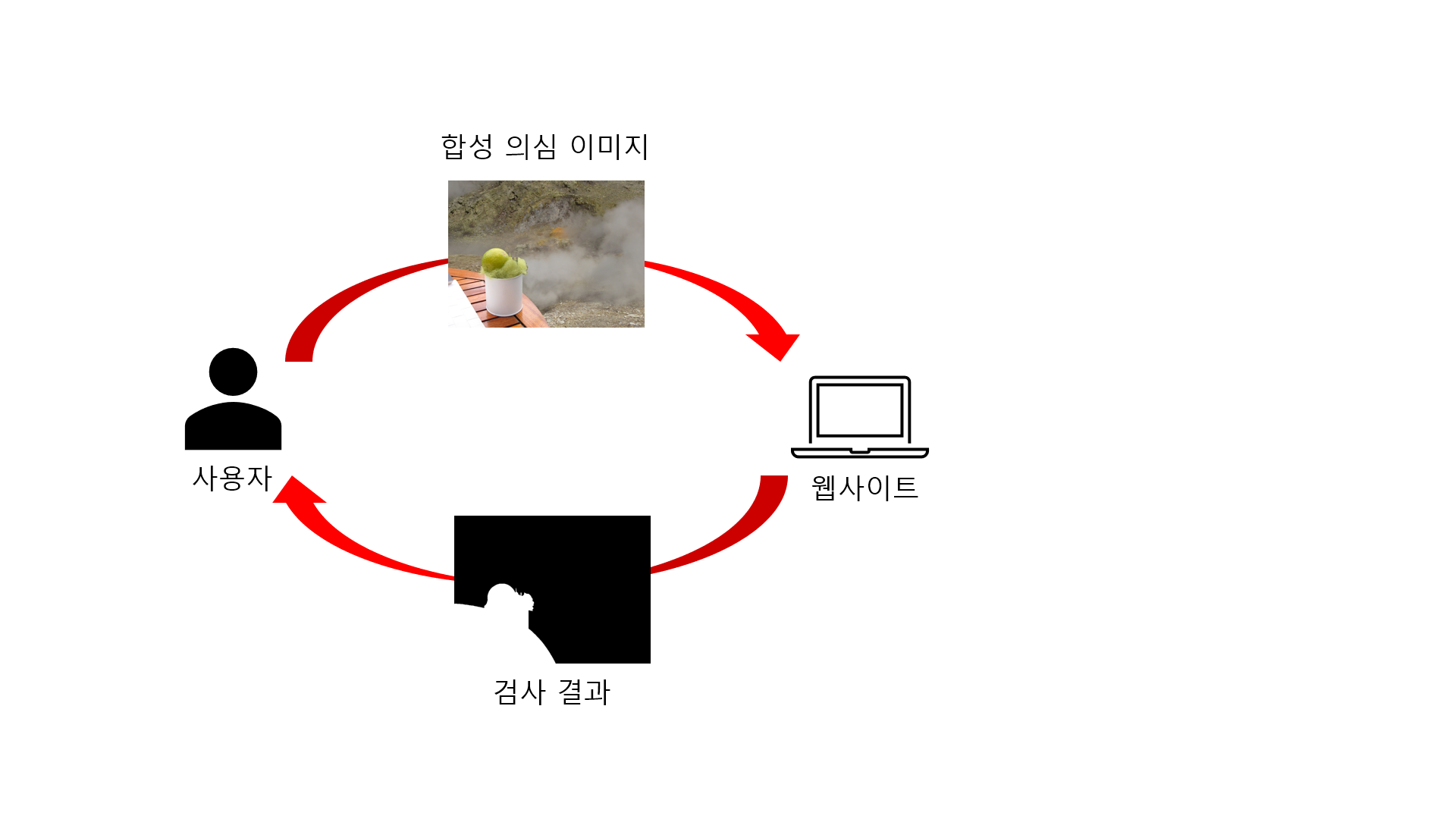


그림 4. 합성 판별기의 개념도

먼저 사용자가 합성으로 의심스러운 사진, 혹은 합성인지 알고 싶은 이미지를 웹사이트에 업로드 한다. 그러면 웹사이트에 업로드된 사진과, 미리 학습시켜놓은 모델을 이용하여 사진에서 합성으로 의심되는 부분을 검출한다. 그 후, 웹사이트를 통해 사용자에게 합성 판별 검사 결과를 웹사이트를 통해 제공한다.

3.2 개발 내용

웹사이트 부분

|  |  |
| --- | --- |
| 기능 | 설명 |
| 사진 업로드 | 사용자는 합성/위조 의심이 가는 이미지를 웹사이트에 업로드 한다. |
| 합성/위조 검사 | 사용자가 웹사이트에 업로드 된 이미지를 미리 학습된 모델을 통해 사용자가 업로드한 사진이 합성(위조)인지 아닌지 판별한다. |
| 검사 결과 제공 | 만약, 이미지가 합성(위조)된 이미지라면, 사용자가 의심 부분을 확인하고 원래 이미지와 비교할 수 있도록 이미지 중 합성(위조)라고 의심되는 부분을 따로 표시하여 “검사 결과”로서 사용자에게 제공한다. |

인공지능 모델 부분

|  |  |
| --- | --- |
| 기능 | 설명 |
| 모델 학습 | 여러 모델을 구현하고 여러가지 조건 하에서 실험을 해본 후, single JPEG compression block과 double JPEG compression block 가장 정확하게 분류 해내는 모델을 선정한다. |
| Localization | 모델을 구축한 후, 모델의 결과값(single JPEG compression block/double JPEG compression block)을 이용하여 합성된 부분을 검출할 수 있도록 코드를 구현한다. |
| Metric | 개발한 Localization의 성능을 측정하기 위해 ground truth와 비교를 진행한다. F1 score와 accuracy를 성능 지표로써 사용한다. |

4. 개발 계획

모든 실험은 Google Colaboratory PRO에서 진행한다.

|  |  |
| --- | --- |
| Info | 사양 |
| RAM | 12.69 GB |
| 디스크 | 147.15 GB |
| CPU | Intel(R) Xeon(R) CPU @ 2.30GHz (cpu family: 6) |
| GPU | Tesla P100 |
| CUDA | 11.2 version |

인공지능, 네트워크와 관련된 전반적인 부분은 파이썬을 활용하며, 웹사이트 제작 부분은 HTML과 Flask를 활용한다. Flask는 파이썬을 활용하여 웹 어플리케이션을 개발 가능하게 해주는 웹 프레임워크이다.

4.1 구현 플랫폼 및 프로그래밍 방법

개발 언어 부분

|  |  |
| --- | --- |
| 개발 언어 | 해당 부분 |
| Python | 데이터 전처리, Network 구성, 성능 평가, Localization 등 Network와 관련된 전반적인 부분  \*\* 3.7.10 버전 이용 |
| HTML | 웹사이트 제작 부분 |

사용한 파이썬 라이브러리

|  |  |
| --- | --- |
| 라이브러리 | 설명 |
| PyTorch | 연구용 프로토타입부터 상용 제품까지 빠르게 만들 수 있는 오픈 소스 머신러닝 프레임워크이다.  \*\* 1.8.1 버전 이용 |
| NumPy | 과학적인 연산(scientific computing)을 위한 기본적인 패키지이다.  다차원 배열 객체, 수학, 논리, 정렬, 선택, 이산 푸리에 변환, 선형 대수 등의 연산을 지원한다. |
| Pillow | Python Imaging Library로, Python 인터프리터에 이미지 처리 기능을 추가해 주는 라이브러리이다.  특히, PIL.JpegImagePlugin에서는 JPEG 형식의 이미지들을 처리한다. |
| Matplotlib | Python에서 통계적, 애니메이션적, 상호적인 시각화를 생성하는 라이브러리이다. |

4.2 진행 상황 체크

Google Colab과 Google 공유 드라이브를 통해 협업하였으며, 매주 화요일 밤 zoom을 통한 정기 회의와 카카오톡 메신저를 활용한 상시 회의를 진행하였다.

|  |  |
| --- | --- |
| 기간 | 개발 목표 내용 |
| 5주차 | 데이터 셋 확보, 데이터 셋 확인, 타겟 논문 확인 |
| 6주차 | 데이터 셋 확보, 데이터 EDA, 네트워크 구성 |
| 7주차 | 데이터 셋 확보, 네트워크 구성(시험기간) |
| 8주차 |
| 9주차 | 데이터 셋 확보, 네트워크 구성, 성능 비교 |
| 10주차 | 네트워크 구성, 성능 비교 |
| 11주차 | 테스트 이미지 적용, 성능 비교, 모델 퍼포먼스 향상  - 모델 구조(RNN/GRU/LSTM 등) 바꾸어 가면서 모델 성능 비교 |
| 12주차 | 테스트 이미지 적용, 성능 비교, 모델 퍼포먼스 향상  - 하이퍼 파라미터 변경해가면서 모델 성능 비교 |
| 13주차 | 테스트 이미지 적용, 성능 비교, 모델 퍼포먼스 향상  - 모델 4개 선택 후 localization에 적용  - Localization 후처리 적용 |
| 14주차 | 지금까지 했던 실험 내용들을 정리하고 최종 발표 진행 |
| 15주차 | 코드 정리 및 최종 보고서 작성 |

5. 개발한 세부 기술

5.1 주요 기능

|  |  |
| --- | --- |
| 기능 | 설명 |
| 웹사이트 | 사용자에게 보여지는 사이트이며, 사용자는 웹사이트를 통해 자신이 합성이라고 의심하는 사진을 업로드한 후 해당 사진의 검사 결과를 확인할 수 있다. |
| 데이터셋 제작 | 인공지능 모델을 train/test하기 위하여 직접 데이터셋을 제작하였다. RAW이미지를 압축할 때에 여러 종류의 quantization table을 사용하여 이미지를 압축하였다. |
| 네트워크 구축 | 합성 판별에 사용하기 위해 해당 JEPG block이 single compression block인지 double compression block인지 구별할 수 있는 모델을 구현하였다. |
| Localization | 미리 학습한 모델의 결과를 이용하여 합성 의심 부분을 검출한다. |

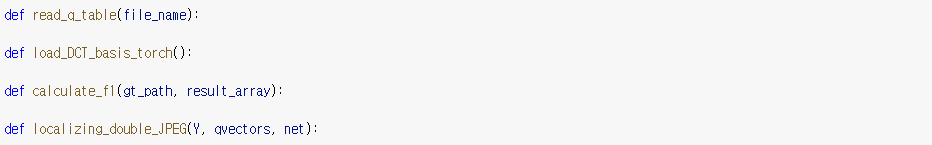
5.2 소스코드 및 설명

5.2.1 웹사이트

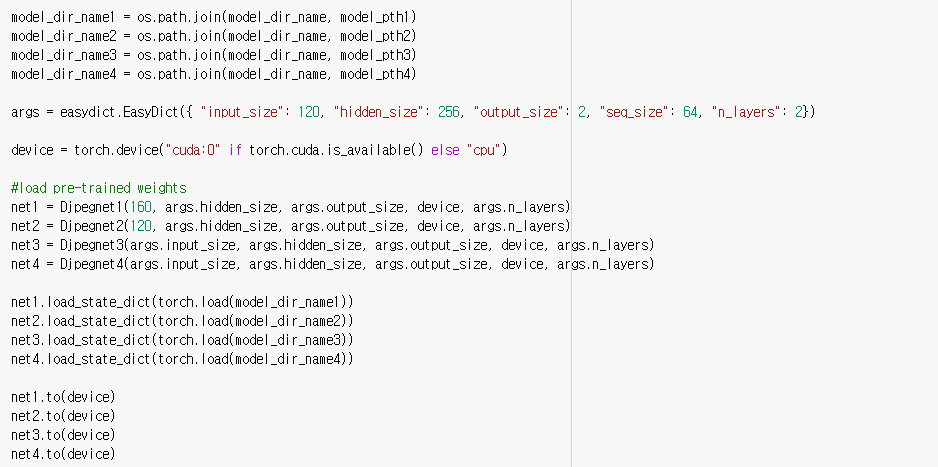
웹사이트는 총 3개의 페이지로 나누어지며, 먼저 합성이 의심스러운 사진을 업로드 하는 페이지, 합성 판별 결과를 출력하는 페이지, 판별 결과와 GroundTruth를 함께 보여주는 페이지이다.



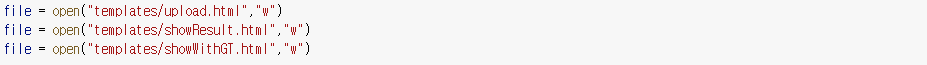
필요한 라이브러리를 임포트하고 필요한 directory를 생성한다.



그 후, 필요한 함수들을 불러오거나 정의한다. 위에서부터 image에서 quantization을 추출하는 함수, DCT 변환을 수행하는 함수, F1 score를 계산하는 함수, localization을 진행하는 함수이다. 이 함수들은 실제 소스코드에서는 각각 Python 파일로 나뉘어져 정의되어 있다.



그 다음 localization에 사용할 모델 4개를 각각 불러온다. 이 모델 4개는 이미 학습되어 있는 모델이며, 각 모델의 하이퍼 파라미터는 아래 네트워크 설명 부분에서 설명한다.

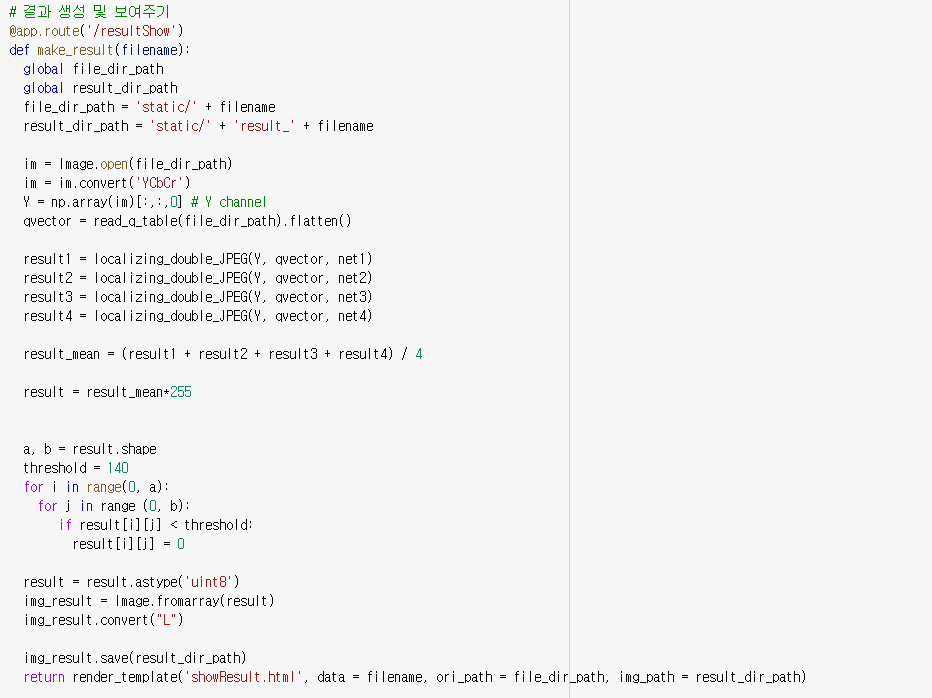


그리고 각각의 웹 사이트 페이지를 HTML으로 작성한다. 위에서부터 차례로 합성이 의심되는 사진을 업로드하는 페이지, 해당 사진에서 합성이 의심되는 부분을 검출하여 보여주는 페이지, Ground Truth가 있는 경우 Ground Truth와 함께 검사 결과를 보여주는 페이지이다.

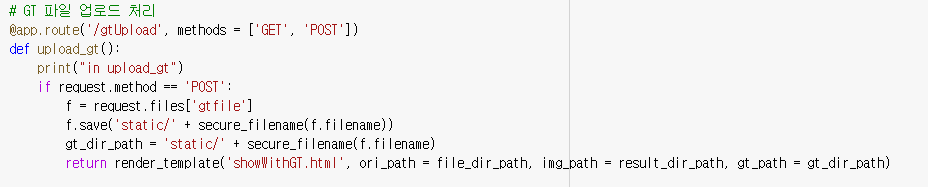


마지막으로 Flask를 활용하여 웹페이지를 구동하는 부분의 코드이다.

파일 업로드 처리 함수에서는 Upload.html 페이지에서 파일이 업로드 되면 업로드 된 파일을 Colab 내의 디스크에 저장하고 make\_result 함수로 이동한다.



make\_result 함수에서는 이미지 사진을 256 x 256 패치로 잘라 4개의 인공지능 모델에 집어 넣고, 검사 결과를 localization한다. 그 후, 4개의 검사 결과를 모두 합쳐 평균을 낸 후, threshold 아래의 값은 0으로 바꾸어주는 후처리를 진행한다. 그 후, 생성된 결과를 showResult.html 페이지에서 보여준다.



만약, showResult.html에서 따로 GroundTruth를 업로드 한다면 showWithGT.html 페이지에서 결과값과 GroudnTruth 이미지를 함께 보여준다.

5.2.2 데이터셋

5.2.2.1. RAW image

(1) RAISE dataset

Digital forgery detection algorithm의 evaluation 지표로 많이 사용되는 real-world image dataset이다. 3개의 카메라 모델로 촬영한 총 8,156개의 RAW image 중 1,000개를 사용하였다. 이미지는 7개의 태그(outdoor, indoor, landscape, nature, people, object, buildings)로 분류되어 있고, 무작위로 선별하였다. 이미지의 크기는 4288\*2848, 4928\*3264 두 가지로 이루어져 있다. 우리 네트워크의 input으로 사용하기 위해, 1,000개의 사진에서 각각 10개의 patch를 추출하여, 총 10,000개의 이미지 패치를 사용하였다. RAW image를 256\*256의 patch block으로 겹치지 않게 나누고, 그 중 10개의 patch block을 무작위로 선별하였다. 패치 크기로 정확히 나누어 떨어지지 않는 부분은 사용하지 않았다.

(2) ALASKA#2

Deep Learning techniques for steganalysis and steganography의 데이터셋으로 사용되는 real-world image dataset이다. 40개 이상의 카메라 모델로 촬영한 총 80,000개의 RAW image 중 20,000개를 사용하였다. 이미지의 크기는 256\*256 단일 사이즈로 이루어져 있다. 우리 네트워크 input 사이즈와 일치하기 때문에 RAW 이미지에서의 추가적인 조작은 없었다.

5.2.2.2. Quantization table

타겟 논문의 저자는 2년간 포렌식 사이트를 운영하며 127,874 장의 image를 확보했고, 그 중 99,677장의 JPEG 이미지에서 standard quantization table을 포함하여 1,170개의 quantization table을 추출해냈다. 많은 image manipulation software에서 nonstandard quantization table을 사용하기 때문에, 범용성을 위해 데이터셋 제작 시 nonstandard quantization table도 사용하였다.

테이블이(가) 표시된 사진

자동 생성된 설명

5.2.2.3 Single compressed JPEG, Double compressed JPEG

ALASKA dataset의 이미지 20,000개를 한 이미지 당 10개의 quantization table을 적용하여 총 200,000장의 single compressed JPEG(이하 SJPEG)을 제작하였다. RAISE dataset의 이미지 패치 10,000개를 한 패치 당 1개의 quantization table을 적용하여 총 10,000장의 SJPEG을 제작하였다. DCT 연산은 행렬의 곱, 나눗셈 연산이기 때문에, 동일한 quantization table을 사용하여 한 번 더 압축할 경우, DCT 계수가 완전히 같아지는 문제가 발생한다. 따라서 double compressed JPEG(이하 DJPEG)을 제작할 때에는, 이전의 SJPEG을 만들 때와 다른 quantization table을 적용하여 한 번 더 압축하였다. 따라서 전체 데이터셋 크기는 SJPEG 210,000장, DJPEG 210,000장으로 총 420,000장이다. train:valid:test 비율은 6:2:2이다.

|  |  |  |
| --- | --- | --- |
|  | Single Compression Data | Double Compression Data |
| 개수 | **210,000** | **210,000** |
| Standard : Non-standard | 2:8 | 2:8 |

5.2.3 네트워크 구축

제안하는 네트워크의 전체적인 구조는 다음과 같다.

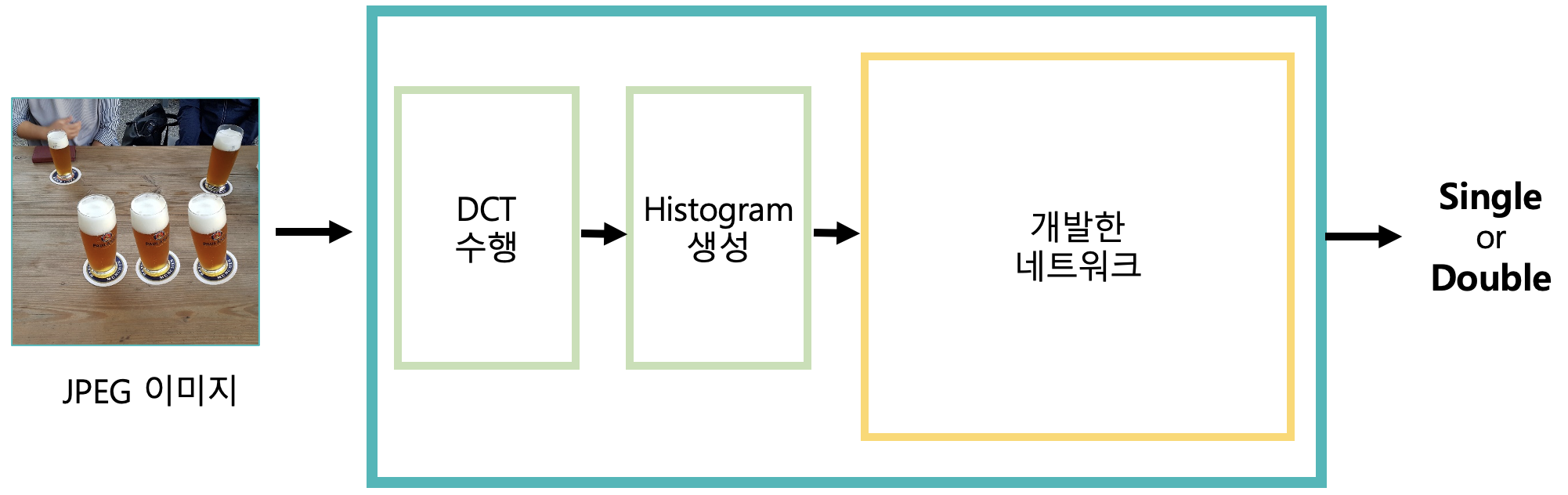


그림 5. 제안하는 네트워크의 전체적인 구조

개발한 네트워크의 자세한 구조는 다음과 같다.

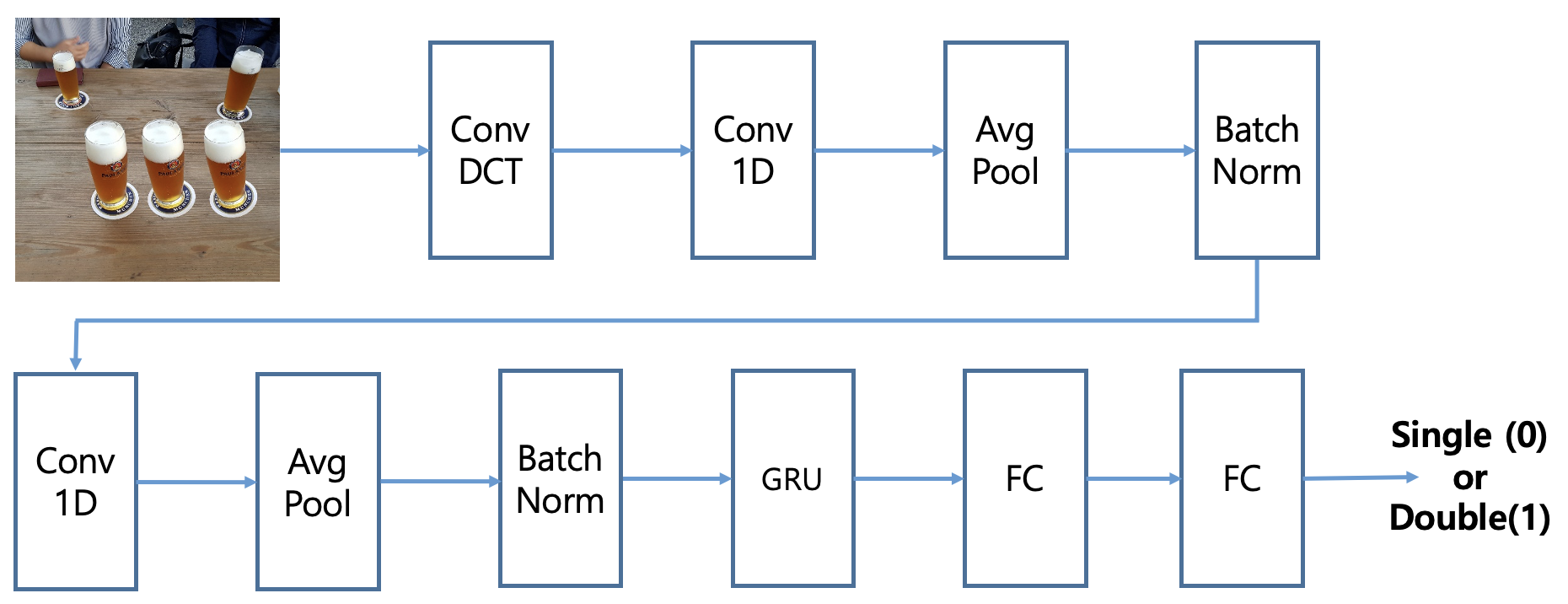
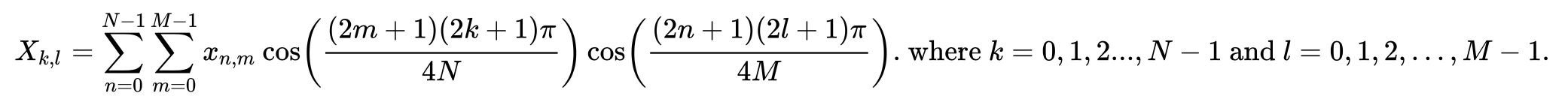


그림 6. 네트워크의 자세한 구조

가. 네트워크의 input - DCT 연산 수행

우선, 주어진 256\*256 크기의 JPEG image의 DCT 연산을 수행한다. 파이썬에서 제공하는 DCT 함수는 1-dimension 연산만 지원하기 때문에, DCT basis를 계산하는 함수를 작성했다. Matrix의 2차원 DCT를 수행하는 식은 다음과 같다.



아래는 소스 코드이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

cal\_scale과 cal\_basis는 각각 DCT 연산에서의 scale 값을 계산하는 함수, DCT basis를 계산하는 함수이다. Scale 값은 p, q가 0일 때 이고, 그 외에는 로 계산하였다. DCT 연산은 8\*8 블록 단위로 진행하므로, N 값은 8이다.

*텍스트이(가) 표시된 사진

자동 생성된 설명*

load\_DCT\_basis\_64() 함수와 load\_DCT\_basis\_torch() 함수는 각각 8\*8 단위로 DCT 연산을 수행하고 pytorch에서 사용할 수 있는 형태로 바꿔주는 함수이다.

나. 네트워크 구조

아래는 네트워크 구조의 소스 코드이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

1D Convolution, Batch Normalization, Average Pooling 연산을 두 번 진행한 다음, GRU 레이어를 거치고, 두 개의 Fully Connected Layer를 통과하면 0 또는 1의 결과 값이 나오는 구조이다. 모든 네트워크에서 hidden state에서의 feature 개수(hidden\_size=256), recurrent layer(n\_layers=2)의 개수는 동일하다. 네트워크마다 input size(bias의 범위), CNN layer의 kernel size와, GRU layer의 bidirectional 여부, dropout 비율과 같은 그 외의 하이퍼 파라미터에는 약간의 차이를 두었다. 아래 표는 각 네트워크에서의 차이점이다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Conv1D-a | Conv1D-b | Bias range | Bidirectional | AC/DC |
| Net1 | Size = 7 | Size = 7 | [-80, 80] | X | Only AC |
| Net2 | Size = 5 | Size = 5 | [-60, 60] | X | Only AC |
| Net3 | Size = 7 | Size = 3 | [-60, 60] | O | Only AC |
| Net4 | Size = 5 | Size = 5 | [-60, 60] | O | AC and DC |

5.2.3 Localization

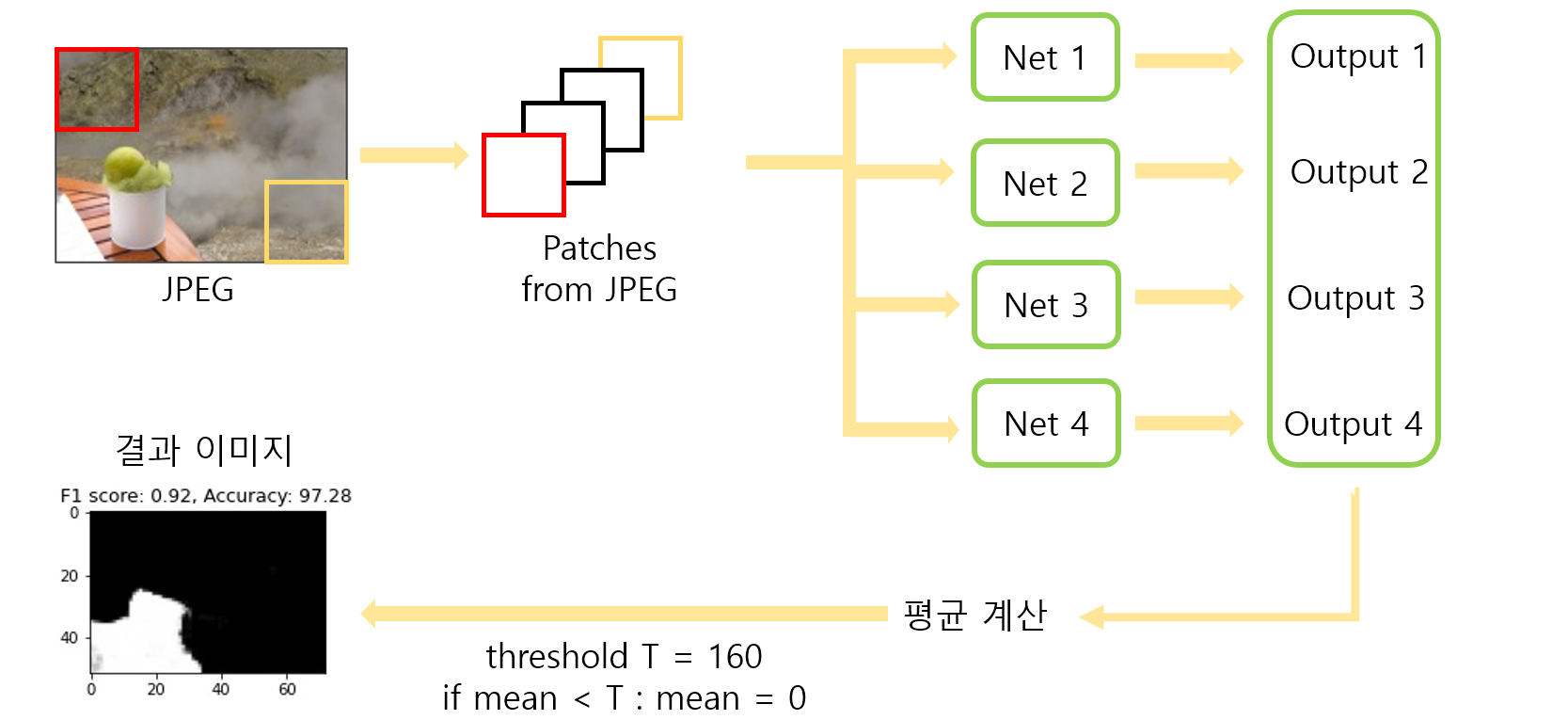


그림 5. Localization 과정

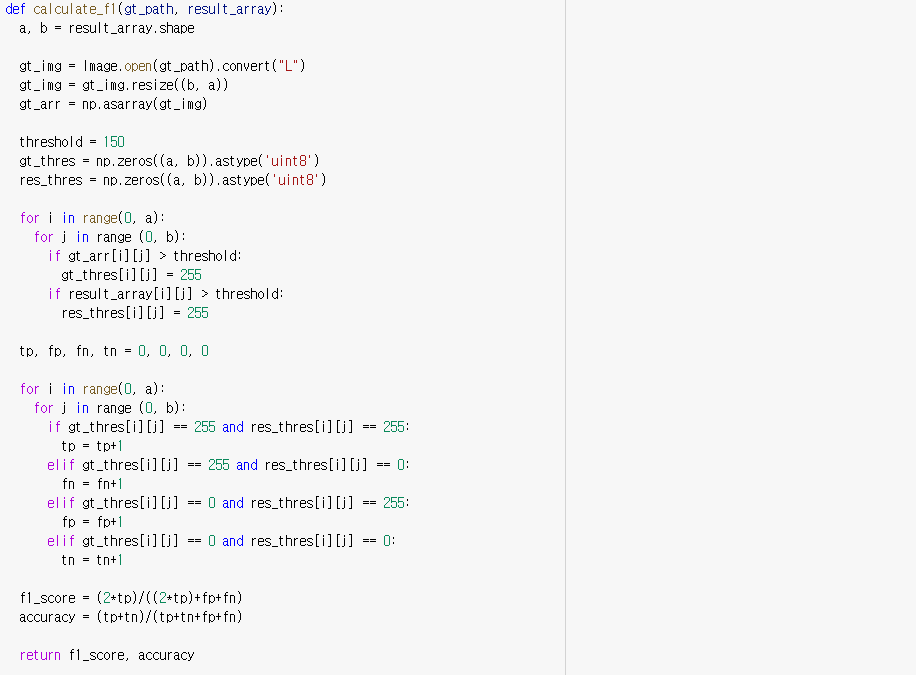
본 프로젝트에서 localization을 수행하는 방식은 다음과 같다. 먼저 합성이 의심되는 이미지에서 256 x 256 크기의 패치를 추출한다. 이때 sliding window 기법을 활용하며, stride는 32로 설정하여 겹치는 부분이 생기도록 한다. 그 후, 훈련된 4개의 모델이 이미지 패치를 input으로 넣는다. 이때, 모델의 output은 각각 모델에서 예측하는 해당 block이 single compression JPEG block일 확률이다. 이 확률을 이미지처럼 array로 쌓아 각각 모델마다 array를 하나씩 만든다. 그 후, 이 4개의 array를 평균 낸다. 그 후, threshold 값을 160으로 두고, 이 값보다 아래의 값은 0으로 만든다. 이것은 single compression block일 확률이 적은 부분을 까맣게 만들어 보다 더 명확하게 합성 부분이 나오게 하는 효과를 준다. 소스 코드 설명은 아래와 같다.



먼저 이미지의 패치를 뽑는 함수를 정의한다.



그 다음 모델 1개에 이미지 패치들을 집어넣으면서 그 결과들을 array로 쌓는 localizing\_double\_JEPG 함수를 정의한다.



그 다음 ground truth와 비교하여 localization의 성능을 측정하기 위해 F1 score와 Accuracy를 측정하는 함수를 정의한다. ground truth와 모델의 result array를 보면 값이 0부터 255 사이로 매우 다양하기 때문에, 일정 threshold를 두어 이 값보다 높다면(합성이라고 명확하게 판별되는 부분이면) 모두 255로 바꾸어 주는 과정을 추가하였다.



그리고 위 웹사이트 부분에서 한 것과 같이, 모델 4개를 불러와 결과를 생성하고, 그 모델 4개를 평균 내어 threshold 값을 이용해 후처리 한다. 그 후, 합성이 의심되는 사진과 모델을 통해 나온 합성 결과 사진, 그리고 F1 score와 Accuracy를 하나의 figure로 만들어 출력한다.

6. 실험 결과 및 고찰

6-1. 기존 논문과의 F1 measure 비교

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method** | Type | ACC(%) | TPR(%) | TNR(%) |
| **VGG-16[14]** | RGB pixels | 50.00 | 00.00 | 100.00 |
| **Wang[3]** | Range – [-5, 5] | 73.05 | 67.74 | 78.37 |
| **Barni[4]** | Range – [-60, 60] | 84.46 | 78.35 | 90.53 |
| **J Park[5]** | Range – [-60, 60] | 92.76 | 90.90 | 94.59 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Our Net1** | Range – [-80, 80] | 91.07 | 87.91 | 94.59 |
| **Our Net2** | Range – [-60, 60] | 92.49 | 89.65 | 95.04 |
| **Our Net3** | Range – [-60, 60] | 91.71 | 88.64 | 94.79 |
| **Our Net4** | Range – [-60, 60] | 92.38 | 88.91 | 95.85 |

Accuracy, True Positive Rate, True Negative Rate 총 세 개의 항목에 대해 localization에 사용한 네 개의 모델(Ours)을 평가하였다. TPR은 double jpeg을 double jpeg이라고 분별한 것, TNR은 single jpeg을 single jpeg이라고 분별한 것이다. 기존 네트워크들과 비교하였을 때, 데이터셋을 대폭 줄인 것에 비해 상당히 준수한 성능을 보였으며, 특히나 single jpeg에 대한 정확도는 타겟 논문을 웃도는 성능을 보였고, double jpeg에 대한 정확도 또한 크게 떨어지지 않았다.

6-2. Localization 결과

|  |  |  |  |
| --- | --- | --- | --- |
| 합성 사진 | Ground truth | ours | 기존 논문 |
| 잔디, 실외, 하늘, 건물이(가) 표시된 사진  자동 생성된 설명 | 실루엣이(가) 표시된 사진  자동 생성된 설명 |  |  |
| 나무, 실외, 잔디, 하늘이(가) 표시된 사진  자동 생성된 설명 | 자연이(가) 표시된 사진  자동 생성된 설명 |  |  |
| 건물, 사람, 실외, 주랑이(가) 표시된 사진  자동 생성된 설명 | 실루엣이(가) 표시된 사진  자동 생성된 설명 |  |  |
| 벽, 노란색이(가) 표시된 사진  자동 생성된 설명 | 실루엣이(가) 표시된 사진  자동 생성된 설명 |  |  |

7. 참고 문헌

1) bskyvision, JPEG 압축 제대로 이해하기, bskyvision, 2019-05-09, https://bskyvision.com/485

2) FILEFORMAT, JPEG – Image File Format, FILEFORMAT, 2021-03-27, https://docs.fileformat.com/image/jpeg/ (게시 날짜가 적혀 있지 않아 검색 일자 표기)

3) Wang, Q., Zhang, R.: Double jpeg compression forensics based on a convolutional neural network. EURASIP Journal on Information Security 2016(1) (2016) 23

4) Barni, M., Bondi, L., Bonettini, N., Bestagini, P., Costanzo, A., Maggini, M., Tondi, B., Tubaro, S.: Aligned and non-aligned double jpeg detection using convolutional neural networks. Journal of Visual Communication and Image Representation 49 (2017) 153–163

5) Park J., Cho D., Ahn W., Lee HK. (2018) Double JPEG Detection in Mixed JPEG Quality Factors Using Deep Convolutional Neural Network. In: Ferrari V., Hebert M., Sminchisescu C., Weiss Y. (eds) Computer Vision – ECCV 2018. ECCV 2018. Lecture Notes in Computer Science, vol 11209. Springer, Cham.

5) PyTorch GitHub, https://github.com/pytorch/pytorch

6) Numpy quickstart, https://numpy.org/doc/stable/user/quickstart.html

7) MatPlotlib GitHub, https://github.com/matplotlib/matplotlib

8) Pillow (PIL Fork) 8.1.2 documentation, https://pillow.readthedocs.io/en/stable/reference/plugins.html

9) WikiDocs, 공학자를 위한 Python - 5.SciPy, https://wikidocs.net/15636

10) Gloe, T., Bo ̈hme, R.: The dresden image database for benchmarking digital image forensics. Journal of Digital Forensic Practice 3(2-4) (2010) 150–15936. (RAW images dataset)

11) Bas, P., Filler, T., Pevny`, T.: break our steganographic system: The ins and outs of organizing boss. In: International Workshop on Information Hiding, Springer (2011) 59–70 (RAW images dataset)

12) Dang-Nguyen, D.T., Pasquini, C., Conotter, V., Boato, G.: Raise: A raw images dataset for digital image forensics. In: Proceedings of the 6th ACM Multimedia Systems Conference, ACM (2015) 219–224 (RAW images dataset)

13) Schaefer G, Stich M. UCID: an uncompressed color image database. Electronic Imaging 2004. International Society for Optics and Photonics. 472-80(2003) (RAW image dataset)

14) Very deep convolutional networks for large-scale image recognition(Simonyan K, 2014)

15) The ALASKA Steganalysis Challenge: A First Step Towards Steganalysis "into the wild", Published in the 7th ACM IH&MMSec conference.

16) Wikipedia, Discrete cosine transform, https://en.wikipedia.org/wiki/Discrete\_cosine\_transform