**졸업논문**

**URL과 이미지 비교를 이용한**

**웹 사이트 위•변조 탐지 서비스**

**Ver. 3.0**

**2019. 06. 23**

**한국외국어대학교 정보통신공학과**

**201403586 허성윤**

**201602119 유진솔**

지 도 교 수 : 정성호

**졸업논문제출 청구 및 승인서**

제 목 : URL과 이미지 비교를 이용한 웹 사이트 위•변조 탐지 서비스

대 학 : 한국외국어대학교 학 과 : 정보통신공학과

학 번 : 201403586 성 명 : 허성윤

학 번 : 201602119 성 명 : 유진솔

이 논문을 제출 하오니 승인하여 주십시오.

2019년 06월 25일

성 명 : 허 성 윤 (인)

성 명 : 유 진 솔 (인)

위 학생의 논문 제출을 승인함.

2019년 06월 25일

지 도 교 수 : 정 성 호 (인)

**문서 정보**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 구 분 | 소 속 | 성 명 | 날 짜 | 서 명 |
| 작성자 | 한국외국어대학교 | 유진솔 | 2019. 06. 23 |  |
| 한국외국어대학교 | 허성윤 | 2019. 06. 23 |  |
| 검토자 | 한국외국어대학교 | 유진솔 | 2019. 06. 23 |  |
| 한국외국어대학교 | 허성윤 | 2019. 06. 23 |  |
| 승인자 | 한국외국어대학교 | 정성호 | 2019. 06. 23 |  |

**머리말**

본 논문에서는 URL과 이미지 비교를 이용한 웹사이트 위•변조 탐지 기법을 제안한다. 제안하는 시스템은 웹사이트 위•변조로 인한 개인정보 유출을 사전에 방지하기 위해 User가 접속한 URL과 웹사이트의 이미지를 추출하고 기존 웹사이트와의 유사도를 비교하여 위•변조 여부를 판단 후 User에게 판단 결과를 제공해 준다.

**개정 이력**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **버전** | **작성자** | **개정일자** | **개정 내역** | **승인자** | | |
| 1.0 | 허성윤 | 2019. 03. 31 | 초안 작성 |  | | |
| 유진솔 |
| 검토자 | 허성윤, 유진솔 | | | | |
| 1.1 | 허성윤 | 2019. 04. 01 | 초안 수정 |  | | |
| 유진솔 |
| 검토자 | 허성윤, 유진솔 | | | | |
| 1.2 | 허성윤 | 2019. 04. 02 | 제안서 최종 수정 | | |  |
| 유진솔 |
| 검토자 | 허성윤, 유진솔 | | | | |
| 2.0 | 허성윤 | 2019. 05. 19 | 본론 작성(1) | |  | |
| 유진솔 |
| 검토자 | 허성윤, 유진솔 | | | | |
| 3.0 | 허성윤 | 2019. 06. 23 | 본론 작성(2) 및 최종 완성 | |  | |
| 유진솔 |
| 검토자 | 허성윤, 유진솔 | | | | |

목 차

목차

[목 차 6](#_Toc12218080)

[1. 서론 12](#_Toc12218081)

[1.1 서비스 정의 12](#_Toc12218082)

[1.2 서비스 기획 배경 및 목적 12](#_Toc12218083)

[1.2.1 웹사이트 위•변조 공격 – 피싱(Phishing)과 파밍(Pharming) 12](#_Toc12218084)

[1.2.2 웹사이트 위•변조 공격 – hosts 파일 및 hosts.ics 파일 변조 13](#_Toc12218087)

[2. 관련 연구 조사 및 분석 14](#_Toc12218091)

[2.1 Feature Matching 기반 탐지 기법 14](#_Toc12218092)

[2.2 URL 기반 탐지 기법 14](#_Toc12218095)

[2.3 시각적 추출 기법(Visual Descriptor)과 SVM을 이용한 탐지 기법 15](#_Toc12218102)

[3. 주제 제안 16](#_Toc12218103)

[3.1 시나리오 16](#_Toc12218104)

[3.1.1 URL 일치 여부 확인 16](#_Toc12218105)

[3.1.2 웹 크롤링을 이용한 웹사이트 이미지 저장 16](#_Toc12218106)

[3.1.3 Inception-v3를 이용한 이미지 유사도 검출 17](#_Toc12218107)

[3.1.4 dHash를 이용한 이미지 유사도 검출 17](#_Toc12218108)

[3.2 사용 기술 설명 17](#_Toc12218109)

[3.2.1 TensorFlow 17](#_Toc12218110)

[3.2.2 Transfer Learning 18](#_Toc12218111)

[3.2.3 dHash Algorithm 18](#_Toc12218112)

[4. 시스템 구성 20](#_Toc12218113)

[4.1 시스템 구성도 20](#_Toc12218114)

[4.2 시스템 동작 순서 21](#_Toc12218115)

[4.2.1 URL 비교 22](#_Toc12218117)

[4.2.2 이미지 전처리 23](#_Toc12218120)

[4.2.3 이미지 학습 25](#_Toc12218123)

[4.2.4 이미지 유사도 비교 27](#_Toc12218124)

[4.2.5 위•변조 판별 30](#_Toc12218125)

[5. 결론 33](#_Toc12218138)

[6. 개발 일정 34](#_Toc12218139)

[7. 참고 문헌 35](#_Toc12218140)

[8. 부 록 36](#_Toc12218141)

1. 서론

1.1 서비스 정의

본 서비스는 URL과 이미지 비교를 이용하여 공공기관 및 금융기관 웹 사이트의 위•변조 여부를 탐지하여 사용자에게 알림 까지 제공하도록 응용 프로그램을 제공한다.

1.2 서비스 기획 배경 및 목적

온라인을 이용한 업무와 금융서비스의 보편화로 인하여, 사용자가 늘어남에 따라 정부나 금융기관 사이트를 모방해 개인•금융 정보를 노리는 해커들로부터 쉽게 노출되는 상황이 발생하게 되었다. 정부나 금융기관 사이트를 모방해 개인•금융 정보를 노리는 피싱(phishing: 개인정보를 빼내는 범죄) 사이트 차단 건수가 2017년 기준 처음 1만건을 넘어선 것으로 나타났다. 피싱 사이트는 금감원이나 은행 사이트와 똑같이 위장해 사용자들의 개인정보를 노린다.

한국인터넷진흥원(KISA)에 따르면 국내 웹사이트 변조 해킹이 2015년 615건, 2016년 1,056건(전년 대비 72%증가), 2017년 1,724건(전년 대비 63%증가)로 매넌 60% 이상 급증하고있는 추세이다. 이와 같은 웹사이트 위•변조 는 사용자가 사이트의 위•변조 여부를 직접 판단하기 어렵기 때문에 피해가 많이 발생하고 있는 실정이다.

이러한 웹사이트 위•변조 문제를 예방하기 URL과 웹사이트의 이미지를 이용하여 효율적으로 웹사이트를 탐지하고 예방할 수 있는 시스템을 제안한다.

1.2.1 웹사이트 위•변조 공격 – 피싱(Phishing)과 파밍(Pharming)

피싱(Phishing)은 개인정보(Private Data)와 낚시(Fishing)의 합성어로 웹사이트의 위장 홈페이지를 만든 뒤, 불특정 다수 이메일 사용자에게 메일을 발송하여 위장된 홈페이지로 접속하도록 현혹하여 개인정보를 빼내는 행위를 의미한다.

파밍(Phaming)은 피싱(Phishing)과 조작(Farming)의 합성어로 합법적으로 소유하고 있던 사용자의 도메인을 탈취하거나 DNS(Domain Name System) 이름을 속여서 사용자들이 진짜 사이트로 오인하도록 유도하는 해킹 방법이다.



[그림 1-1]

PC가 파밍 악성 코드에 감염되면, 사용자는 정상 웹사이트에 접속을 했음에도 불구하고, 해커의 의해 변조된 임의의 가짜 사이트로 접속이 되어 개인정보 입력을 유도하게 된다.

1.2.2 웹사이트 위•변조 공격 – hosts 파일 및 hosts.ics 파일 변조

hosts 파일 변조는 사용자가 웹사이트에 접속하기 위해 URL을 입력하면 사용자 시스템의 ‘호스트’ 파일을 확인한다. 파일 내부에 입력된 영문 주소가 있으면 파일에 기록된 IP 주소로 웹사이트에 연결된다. 파일 내부에 영문 주소가 없으면 외부 네트워크에 있는 DNS 서버를 통해 변환된 IP 주소로 웹사이트에 연결된다. 호스트 파일이 변조되면 정상 URL을 입력해도 사용자가 모르는 사이 가짜 사이트로 연결된다.





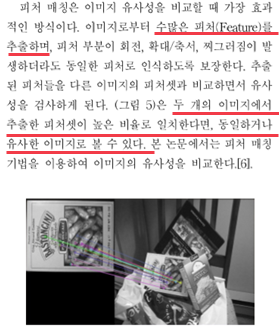
[그림 1-2]

호스트 파일을 변조한 웹사이트가 증가하고 호스트 파일 변조를 각종 보안 프로그램이 탐지하자, 공격자는 새로운 파일을 변조하게 됐다. 바로 hosts.ics 파일 변조이다. 웹사이트에 접속할 때 ‘hosts’ 파일보다 ‘hosts.ics’ 파일을 먼저 확인하게 된다.

따라서 보안 프로그램에 의해 변조 여부를 감시 받는 호스트 파일을 변조하지 않고, hosts.ics 파일을 변조하여 사용자를 가짜 사이트로 유도하는 것이다.

1. 관련 연구 조사 및 분석

2.1 Feature Matching 기반 탐지 기법

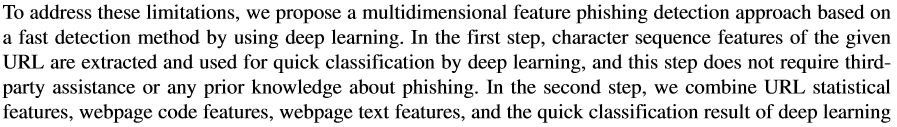


[그림 2-1]

Feature Matching이란 수많은 Feature를 추출하여 다른 이미지의 Feature Set과 비교 후 유사성을 검사하는 OpenCV의 일종이다. 이미지에서 검출한 특성 포인트에 대해 색상을 지정해서 표시하고, 일치하는 특성 포인트를 하나의 선으로 연결한다. 특성 디스크립터의 개수를 줄이거나 늘임으로써 선의 개수를 조절할 수 있다. 하지만 선의 개수를 늘린다고 해도 이미지 전체에 대한 특성을 추출해내는 데에 한계가 있고, 그 특징을 추출하는 데에 시간이 상당히 소요되기 때문에 유사도 정확성과 효율이 떨어진다.

2.2 URL 기반 탐지 기법





[그림 2-2]

위 논문은 URL을 기반으로 위•변조 웹 사이트를 탐지한 연구이다. 이 연구에서는 URL 시퀀스

기능을 추출하여 Deep Learning을 통해 분류하는 로직으로, URL 통계 기능, 웹 페이지 코드 기능, 웹

페이지 텍스트 기능 및 Deep Learning 분류 결과를 이용한다. 그러나, 이는 단지 URL만 탐지에 사용하기 때문에 정상 도메인 주소를 가진 피싱 사이트 탐지에 부적합하다는 결점이 있다.

2.3 시각적 추출 기법(Visual Descriptor)과 SVM을 이용한 탐지 기법



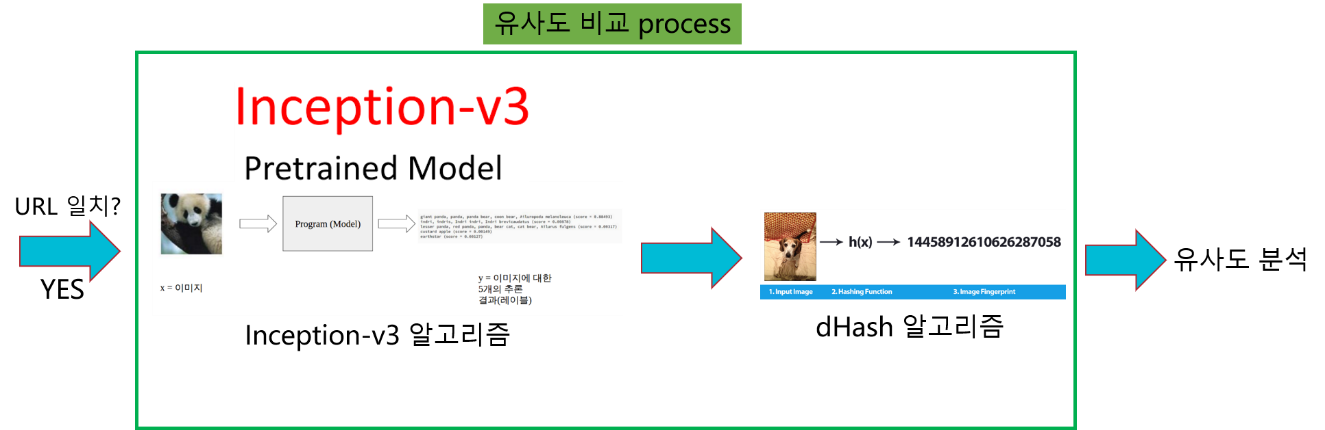


**[그림 2-3]**

위 연구에서는 이미지의 특징을 추출하는 데에 Visual Descriptor를 사용해서 이미지의 색상이나 질감 등의 시각적 특징만을 추출해 낸다. 이후에 이미지를 분류하는 과정에 SVM을 사용한다. 하지만, Visual Descriptor는 이미 오래 전 기술이고 단순히 이미지의 시각적 특징만을 추출해 내기 때문에 다중 필터를 사용하는 Inception-v3의 성능이 더 우수하다고 판단하였다. 또한, 일반적으로 SVM은 본 연구에서 이용하는 Inception-v3에 비해 정확성이 떨어진다.

1. 주제 제안

3.1 시나리오



**[그림 3-1]**

접속한 웹 페이지의 URL과 오리지널 URL의 일치 여부를 확인하고 일치 한다면, Inception-v3로 학습시켜 둔 모델에 비교 대상 이미지를 입력으로 넣어준다. 그렇게 해서 나온 결과 유사도 값과 dHash를 이용한 이미지 유사도를 분석하여 적당한 가중치를 준 뒤 이러한 과정을 적용하여 더 정확한 이미지 유사도를 검출하고자 한다. 본 논문에서는 어느 한 은행의 모든 웹 페이지를 캡쳐 하여 그것을 Input Data로 넣어주어 학습시킨다. 이러한 하나의 case를 제시하여, 이후에 다른 은행 및 웹 사이트에도 같은 기술이 적용될 수 있음을 보인다.

3.1.1 URL 일치 여부 확인

* 1. 프로그램을 통해 사용자의 URL을 입력 받는다..
  2. User가 접속 중인 웹 사이트 화면을 Python selenium으로 캡쳐한 이미지 이름을 해당 URL로 저장한다.
  3. 사용자가 입력한 URL이 Original URL list에 있는 URL과 일치하는지 비교한다.
  4. 일치하지 않으면 위조사이트로 판단하고, 일치한다면 이미지 유사도 비교를 시작한다.

3.1.2 웹 크롤링을 이용한 웹사이트 이미지 저장

1. 기존 사이트(Original site)의 URL을 받는다.
2. Python BeautifulSoup을 이용하여 해당 URL의 html에서 다른 링크로 갈 수 있는 URL만을 추출한다.
3. 추출한 URL들 중 https로 시작하는 유효한 URL만 다시 추출한다.
4. 유효한 URL에 접속 후 이 URL의 html에서 다른 링크로 갈 수 있는 URL들을 추출하여 저장해 둔다.
5. 이후 Python Selenium을 이용하여 접속한 웹사이트 이미지를 캡쳐하고 해당 이미지의 URL이름으로 저장한다.
6. 이미지를 캡쳐할 때마다 그 이미지의 URL을 텍스트 파일(Original URL list)에 저장한다.
7. 사이트에서 새로고침 할 때마다 다른 이미지로 바뀌는 이미지들이 존재하므로 1초 간격으로 새로고침하여 한 URL당 총 20장의 이미지를 저장한다.

3.1.3 Inception-v3를 이용한 이미지 유사도 검출

1. 오리지널 사이트의 이미지 데이터 셋을 만든다.
2. 오리지널 사이트의 이미지를 URL별로 폴더로 분류하고 이 폴더를 데이터 셋으로 사용한다.
3. 하나의 폴더에 새로고침하여 저장한 총 20개의 이미지가 존재한다.
4. Inception V3를 사용하여 이미지 학습을 한다.
5. 오리지널 사이트의 데이터 셋을 input으로 넣어주고 이 이미지들을 (n, 299, 299 3)의 배열로 변환한다.
6. Convolution Layer와 Pooling Layer과정을 거쳐서 (n, 2048)의 배열로 변환한다.
7. Fully connected로 m개의 class로 분류한다.
8. Transfer Leaning을 위한 기존의 이미지 학습 모델의 오픈소스인 retrain.py를 이용하여 이미지 학습을 하여 Label을 생성한다.
9. 학습된 결과로 나온 Output Label를 이용하여 유사도를 검출한다.

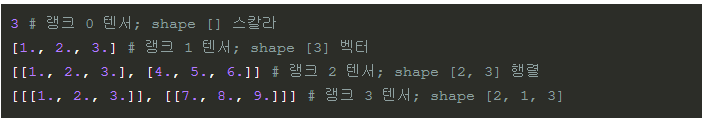
3.1.4 dHash를 이용한 이미지 유사도 검출

1. 오리지널 사이트의 이미지와 사용자가 접속한 사이트의 이미지를 비교하여 유사도를 검출한다.
2. Python에서 제공하는 libarary인 ImageHash에서 처리 속도가 빠르고 이미지 비교에 적합한 dHash 알고리즘을 사용한다.
3. 결과값으로 두 이미지의 차이 정도를 나타내는 해밍 거리 값을 얻는다.
   1. 검출된 유사도 정보로 사용자가 접속하고있는 사이트에 대한 유사도 정보를 제공한다.
4. 결과 해밍 거리 값이 0이라면 두 이미지가 동일한 이미지라고 판단한다.
5. 이 정보를 이전의 Inception V3를 통한 이미지 유사도 비교 결과와 종합하여 프로그램을 통해 User에게 이 웹사이트의 위,변조 여부를 최종적으로 알려줄 수 있다.

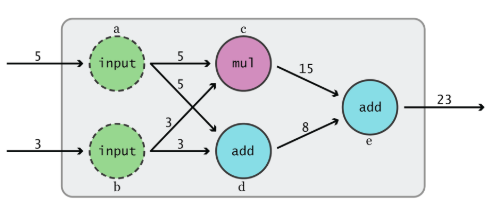
3.2 사용 기술 설명

3.2.1 TensorFlow

텐서플로우는 데이터 흐름 그래프를 기반으로 하는 수치 계산을 위한 소프트웨어 프레임 워크이다. 텐서(tensor)는 Multidimensional Arrays라고 하며, 딥러닝 에서의 텐서는 다차원 배열로 나타내는 데이터를 의미한다. 랭크는 행렬들의 열들로 생성될 수 있는 백터 공간의 차원을 의미한다.



**[그림 3-2]**

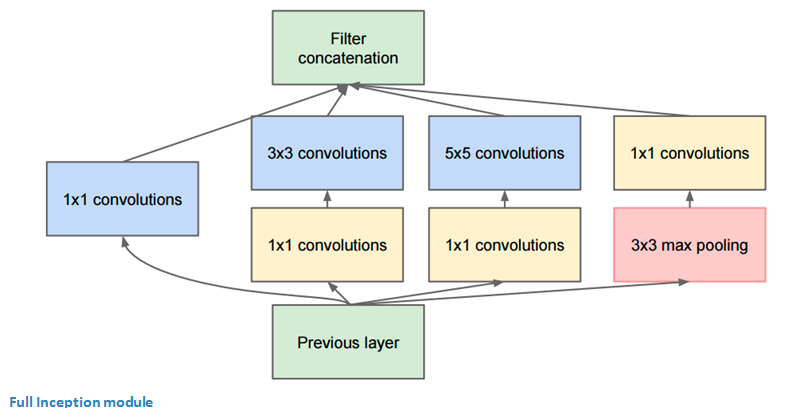


**[그림 3-3]**

텐서는 노드(node)와 엣지(edge)로 구성된 Data Flow Graph를 통해 연산을 수행한다.

3.2.2 Transfer Learning

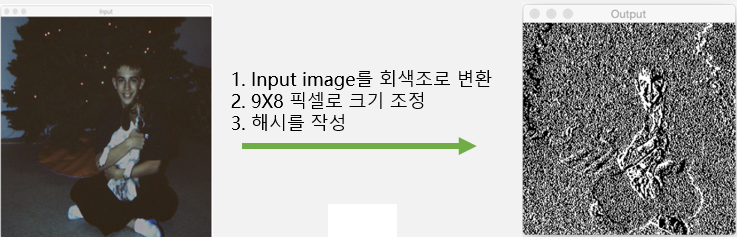
기존의 만들어진 모델을 사용하여 새로운 모델을 만들 시 학습을 빠르게 하며, 예측력을 높이는 방법이다.

**[그림 3-4]**

Transfer Learning은 실질적으로 Convolution network을 처음부터 학습시키는 경우는 드물기 때문에, 이미 학습된 모델을 사용하여 문제를 해결하는데 이용한다.

세부 조정(Fine-tuning)을 사용하여 기존 학습된 모델의 특징 추출과 다른 layer를 고정시키고 일부 layer만 조정하여 재 학습을 한다.

3.2.3 dHash Algorithm

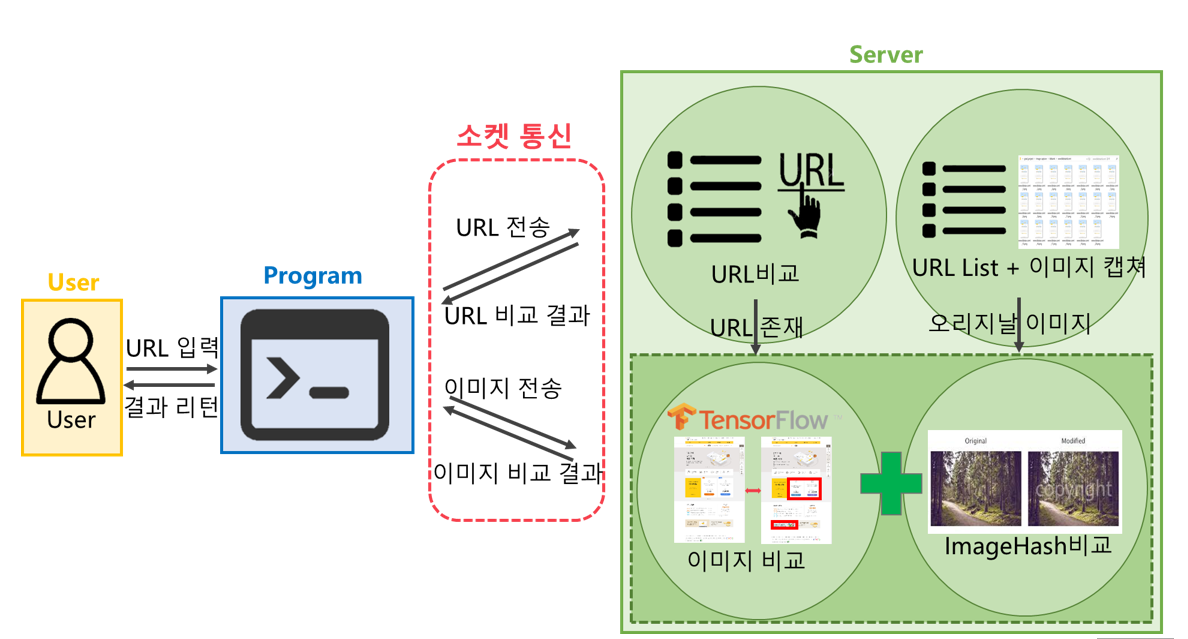


**[그림 3-5]**

dHash 알고리즘은 이미지의 인접 픽셀 간의 차이를 계산하여 만들어진 64비트 해시를 이용하여 해밍 거리를 구하고, 해밍 거리 값으로 이미지 유사도를 결정하는 방법이다.

1. 입력이미지를 회색조로 변환한다.
2. 이미지 크기 조정을 9 x 8 픽셀 나노로 크기 조정을 하여, 결과 이미지 해시가 초기 공간 크기에 관계없이 비슷한 사진과 크기가 일치 하도록 한다.
3. 인접한 픽셀의 차이를 계산하면 8가지 차이가 생겨서 8x8의 64비트 해시가 되고, 이 비트를 할당하여 해시를 작성한다.
4. 서로 다른 두 해시의 비트를 비교하여 해밍거리를 구한다.
5. 해밍거리가 0이면 두 해시가 동일하다, 즉 두 이미지가 동일하다는 것을 의미한다.
6. 시스템 구성

4.1 시스템 구성도



**[그림 4-1]**

1. User가 프로그램을 통해 URL을 입력하면 URL이 소켓 통신을 통해 서버로 전송된다.

2. 서버에서는 프로그램으로부터 받은 URL이 이전에 미리 저장해둔 Original URL list에 있는지 비교한다.

2-1. Original URL list에 없는 URL이라면, 소켓 통신을 통해 프로그램으로 URL 비교 결과를 전송한다.

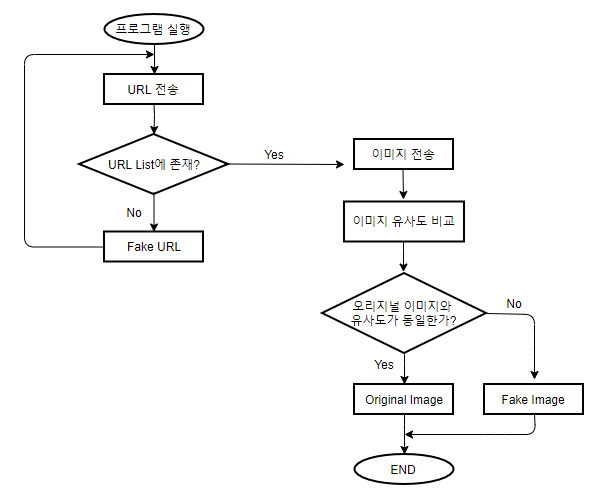
2-2. Original URL list에 있는 URL이라면, 서버는 소켓 통신을 통해 프로그램으로 URL 비교 결과를 전송하고 동시에 이미지를 요청한다. (3으로 이동)

3. 프로그램은 User가 접속해 있는 사이트의 이미지를 캡쳐하여 서버로 이미지를 전송한다.

4. 서버에서는 이미 학습시켜 둔 해당 은행 웹사이트 이미지와 프로그램으로부터 받은 이미지를 Tensorflow(Inception V3)를 이용하여 비교하고 유사도를 검출한다. 또한, 정확성을 높이기 위하여 ImageHash로 한번 더 비교하여 종합적으로 유사도 결과를 낸 후 프로그램으로 이미지 비교 결과를 전송한다.

4. 3의 유사도가 오리지널 이미지의 유사도와 동일하다면 Original Image이므로 프로그램을 통해 User에게 Original Site라는 결과를 보여준다.

4.2 시스템 동작 순서



1. User가 은행 웹사이트에 접속하여 실행 프로그램에서 User가 접속해 있는 URL을 입력하면 URL이 서버로 전송된다.

2. 서버에서는 프로그램으로부터 받은 URL이 이전에 미리 저장해둔 Original URL list에 있는지 확인한다.

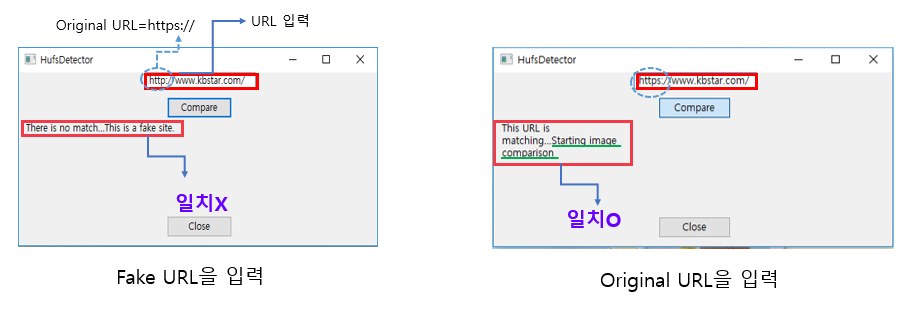
2-1. Original URL list에 없는 URL이라면, 프로그램을 통해 Fake site라는 결과를 User에게 보여준다.

2-2. Original URL list에 있는 URL이라면, 서버는 프로그램으로 이미지를 요청하고 프로그램은 User가 접속해 있는 사이트의 이미지를 캡쳐하여 서버로 전송한다. (3으로 이동)

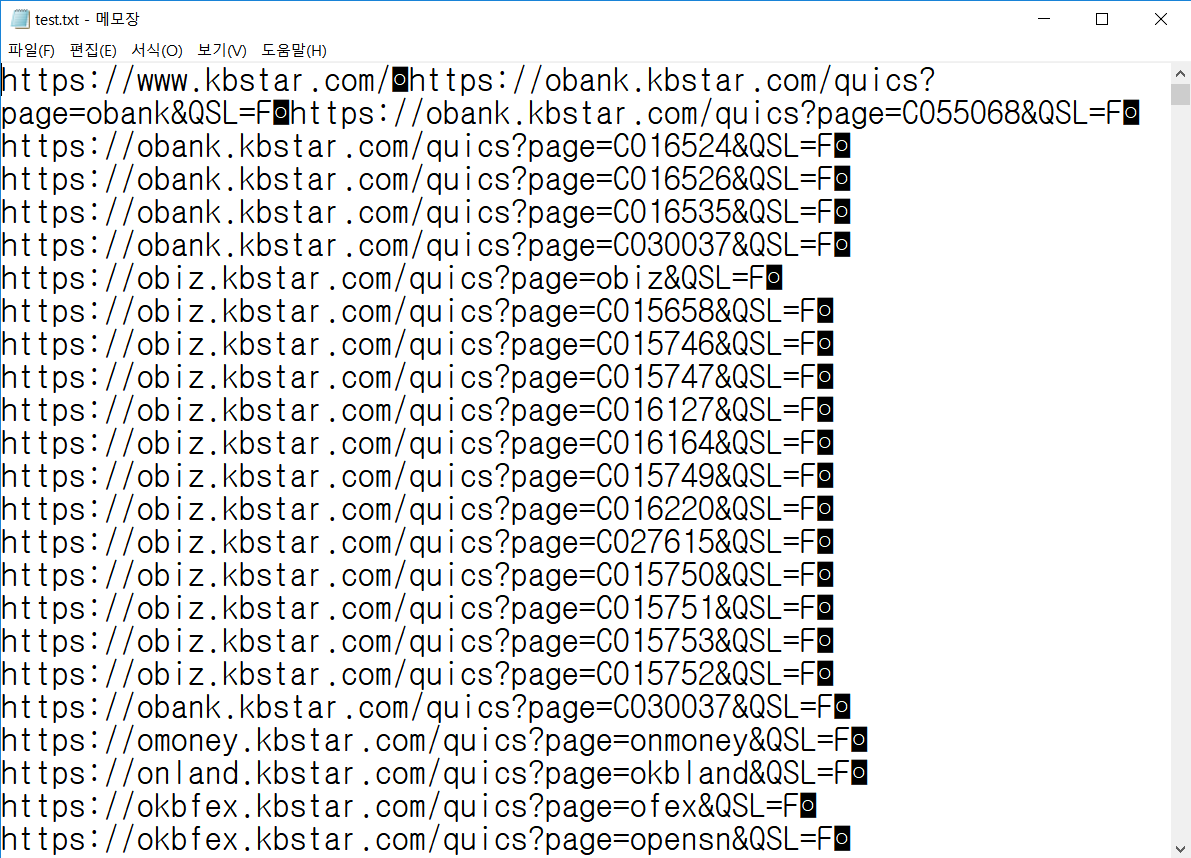
3. 서버에서는 이미 학습시켜 둔 해당 은행 웹사이트 이미지와 프로그램으로부터 받은 이미지를 비교하여 유사도를 검출한다. 또한, 정확성을 높이기 위하여 ImageHash로 한번 더 비교하여 종합적으로 유사도 결과를 낸 후 프로그램으로 전송한다.

4. 3의 유사도가 오리지널 이미지의 유사도와 동일하다면 Original Image이므로 프로그램을 통해 User에게 Original Site라는 결과를 보여준다.

4.2.1 URL 비교



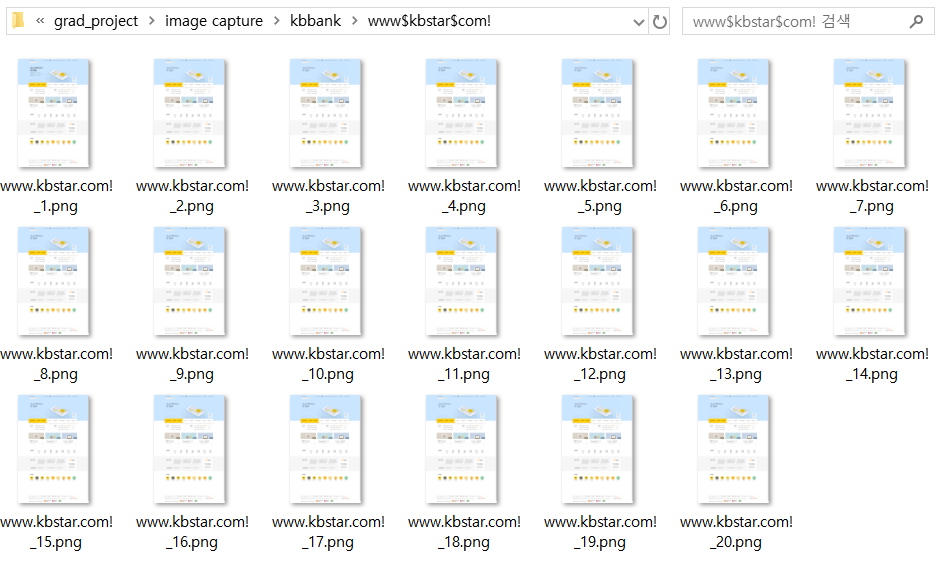
**[그림 4-2]**



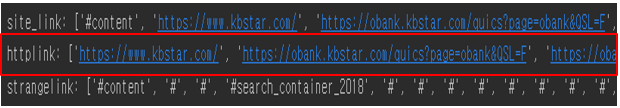
**[그림 4-3]**

[그림 4-2]와 같이 프로그램을 통해 User가 접속한 웹페이지의 URL을 받아올 수 있도록 하였다. [그림 4-3]은 국민은행 웹사이트를 크롤링하면서 동시에 저장한 Original URL list이다. 서버에서는 User로부터 받아온 URL을 이 Original URL list에 저장된 URL들과 비교하여 일치하는 것이 있다면 ‘This URL is matching’이라는 메시지를 프로그램에 전송하고 이미지를 요청한다. 프로그램은 User가 접속한 웹사이트의 이미지를 캡쳐 후 서버에 전송한다. 이후 서버에서는 수신된 이미지와 오리지널 이미지의 비교를 시작한다. 만약, 일치하는 URL이 없다면 ‘There is no match’라는 메시지를 프로그램에 전송한다. 이 때, 서버는 프로그램에 이미지 요청을 하지 않으므로 이미지 비교를 하지 않고 프로그램은 User에게 바로 Fake site라는 결과를 보여줄 수 있도록 동작한다.

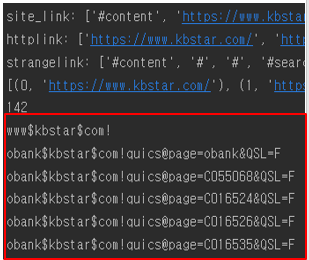
4.2.2 이미지 전처리

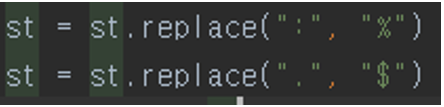


**[그림 4-4]**

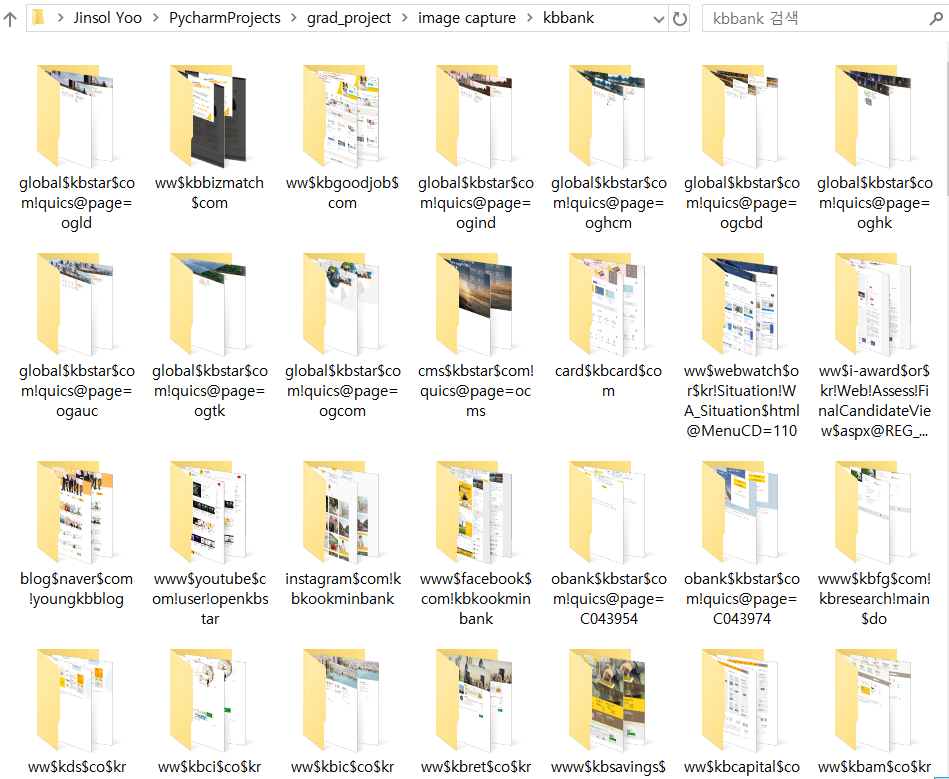


**[그림 4-5]**





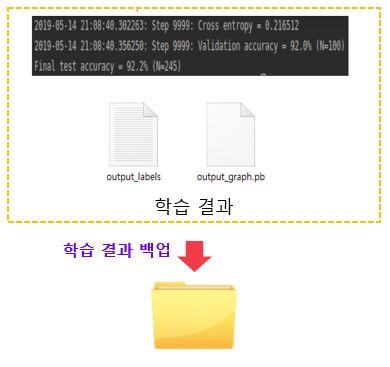
**[그림 4-6]**



**[그림 4-7]**

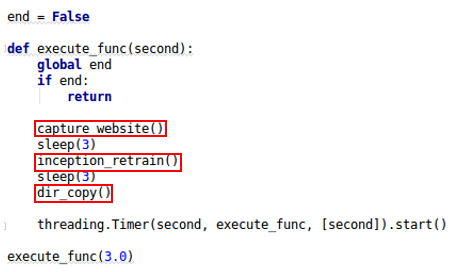
하나의 은행 웹사이트에서 접속할 수 있는 여러 웹페이지들의 이미지까지 최대한 많은 수의 이미지를 캡쳐하기 위해 웹사이트 html에서 현재 링크로부터 다른 링크로 넘어갈 수 있는 URL부분, 즉 <a href>태그 부분의 URL만을 추출한다. [그림 4-5]와 같이 추출된 URL들 중 https로 시작하지 않는 URL들은 제거하고 유효한 URL들만 다시 추출한다. 유효한 URL로 접속하여 캡쳐되는 수많은 이미지들을 구별하기 위해서 [그림 4-4]와 같이 추출한 URL을 이미지 파일 이름으로 하여 저장하였다. 사이트에서 새로고침 할 때마다 다른 이미지로 바뀌는 이미지들이 존재하므로 1초 간격으로 새로고침하여 이미지를 캡쳐한다. 총 20장의 이미지를 캡쳐하여 하나의 URL을 폴더 이름으로 한 폴더에 저장한다. 즉, www.abc.co.kr이라는 폴더에 www.abc.co.kr\_1.png, www.abc.co.kr\_2.png, …의 형태로 이미지들이 저장된다. [그림 4-7]는 그렇게 저장한 결과 이미지 폴더들이다. URL을 폴더 이름으로 하는 하나의 폴더 안에 총 20장의 이미지들이 저장되어 있고, 국민은행 사이트 하나로 코드를 실행했을 때 전체 128개 정도의 폴더가 저장된다. 이 때, Python에서는 파일 이름에 ‘.’이나 ‘:’이 들어갈 경우 이미지 캡쳐가 불가능했기 때문에 [그림 4-6]과 같이 이러한 부분을 다른 특수문자로 바꾸어 저장하였다

4.2.3 이미지 학습



**[그림 4-8]**

[그림 4-8]에서 output\_labels와 output\_graph.pb는 간단히 말해서 이미지 학습이 수행된 후 생성되는 학습 결과라고 볼 수 있다. 즉, 서버에서 User가 위•변조 여부를 판단하고자 하는 이미지를 프로그램으로부터 받았을 때 이 파일들을 이용하여 이미지 비교가 가능하다. 이러한 프로그램의 이미지 비교 요청이 들어왔을 때 서버는 즉시 이미지 비교 수행 후 결과를 프로그램, 즉 User에게 돌려주어야 한다. 그런데 만약 이 때 서버에서 이미지 학습이 동시에 진행 중이라면 output\_labels와 output\_graph.pb 파일 내용이 변경되고 있는 중이기 때문에 이미지 비교가 제대로 이루어지지 않을 수 있다. 따라서, 서버는 이미지 학습이 종료되면 이 파일들을 따로 백업해 둘 필요가 있다. 이렇게 하면, 프로그램으로부터의 이미지 비교 요청이 들어왔을 때 서버는 백업해 둔 파일들을 이용하여 충돌없이 이미지 비교를 수행할 수 있다.



**[그림 4-9]**

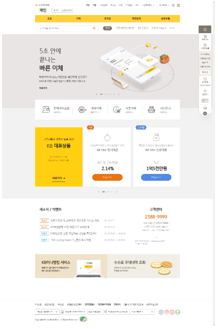
웹 사이트의 갱신 주기가 짧거나 관리자가 그에 따라 이미지 전처리, 학습 등의 과정을 즉각적으로 실행시켜 줄 수 없는 경우가 많기 때문에 서버는 알아서 이미지 전처리, 학습, 백업과정이 끝나면 이 과정을 반복 수행할 수 있어야 한다. [그림 4-9]은 이를 구현한 코드이다. capture\_website()는 이미지를 크롤링하여 캡쳐하는 메소드이고, inception\_retrain()은 이미지 학습을 위한 메소드, dir\_copy()는 학습 결과를 백업하는 메소드이다. 각 메소드는 이전의 메소드가 끝난 이후 3초 후에 실행되고, 3개의 메소드 모두 종료되면 이 또한 3초 후에 다시 이미지 캡쳐 메소드부터 실행하게 된다.

4.2.4 이미지 유사도 비교

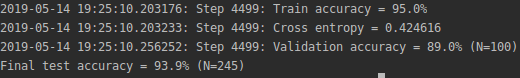


**[그림 4-10]**

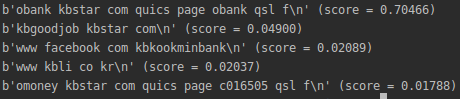
이미지 비교 테스트를 위한 가짜 은행 사이트 이미지를 구하기 쉽지 않았기 때문에 [그림 4-10]과 같이 직접 위조 사이트를 만들었다. 본 논문에서는 실제 국민은행 사이트의 html에서 이미지 부분을 우리은행 이미지로 변경하였고, 도메인 주소도 국민은행의 도메인과 비슷하게 하여 제작하였다.

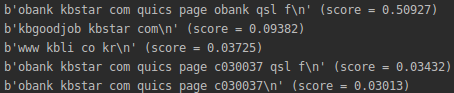
**[그림 4-11] [그림 4-12]**



**[그림 4-13]**

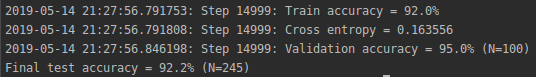


**[그림 4-14]**

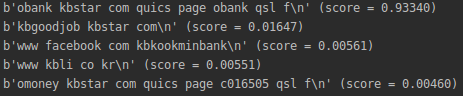


**[그림 4-15]**

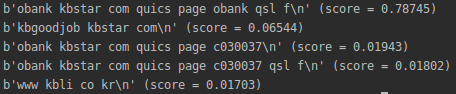
[그림 4-11]은 기존 사이트, [그림 4-12]은 위에서 제작한 위조 사이트이다. 이 이미지들을 가지고 학습 횟수에 따른 유사도를 비교하였다. 앞서 크롤링을 이용하여 저장한 웹사이트 이미지들을 Dataset으로 하여 Inception-v3 모델을 사용해서 이미지 학습을 진행하였다. [그림 4-13]은 4500번 학습시킨 결과, 학습 정확도 93.9%를 보여주고 있다. [그림 4-14]와 [그림 4-15]은 4500번 학습 시킨 후 [그림 4-11]와 [그림 4-12]으로 테스트한 결과이다. [그림 4-14]는 기존 사이트의 유사도 70.5%를, [그림 4-15]은 위조 사이트의 유사도 50.9%를 나타낸다.



**[그림 4-16]**

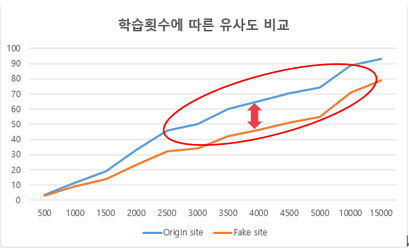


**[그림 4-17]**



**[그림 4-18]**

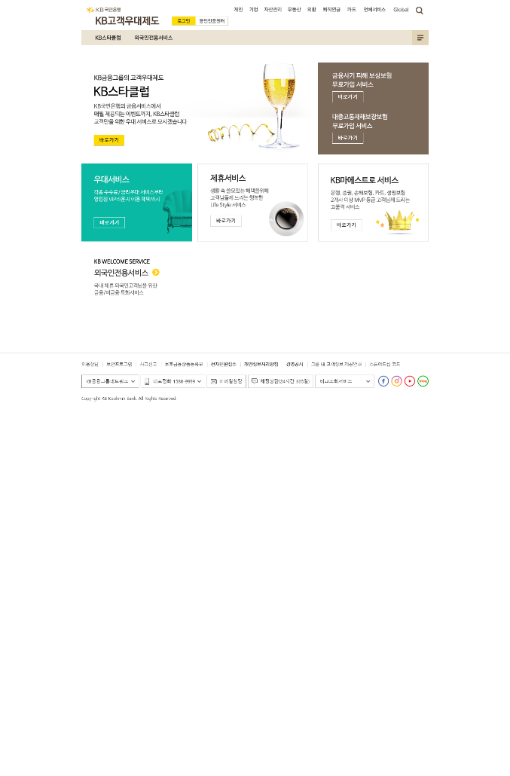
[그림 4-16], [그림 4-17], [그림 4-18]은 학습 횟수를 15000번으로 하고 위에서 사용한 이미지와 같은 테스트 이미지를 사용하여 테스트했을 때의 결과이다. [그림 4-16]는 학습 정확도 92.2%를 보여주고 있고, [그림 4-17]는 기존 사이트의 유사도 93.3%, [그림 4-18]은 위조 사이트의 유사도를 나타낸다.



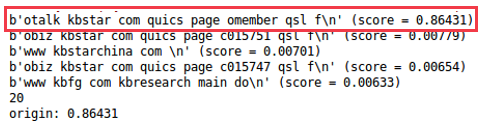
**[그림 4-19]**

[그림 4-19]은 이런 식으로 학습 횟수를 증가시키면서 학습시킨 후 [그림 4-11]와 [그림 4-12]으로 테스트한 결과를 그래프로 나타낸 것이다. 3500번 학습시킨 구간 이후부터 10000번 학습시킨 구간까지 [그림 4-11]와 [그림 4-12]으로 테스트했을 때의 유사도 차이가 약 20% 정도로 Flat해지는 것을 확인할 수 있다. 하지만, 위 그래프에서 3500번보다 학습을 더 적게 시키거나 10000번보다 더 많이 시켰을 때는 유사도 차이가 20%보다 감소한다. 따라서, 학습 횟수를 무한히 늘리거나 줄이지 않고 3500번에서 10000번까지의 구간에서 하나를 기준으로 하여 학습 횟수를 정해야 한다. 본 논문에서는 학습 횟수를 10000번으로 고정시키고 테스트를 진행하였다.

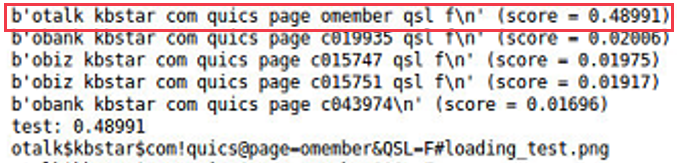
4.2.5 위•변조 판별



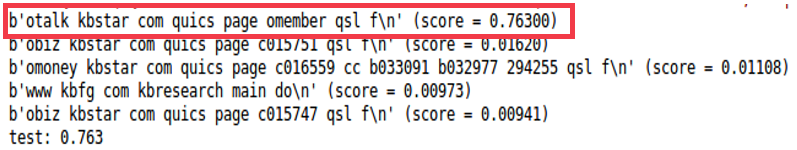
**[그림 4-20] [그림 4-21] [그림 4-22]**



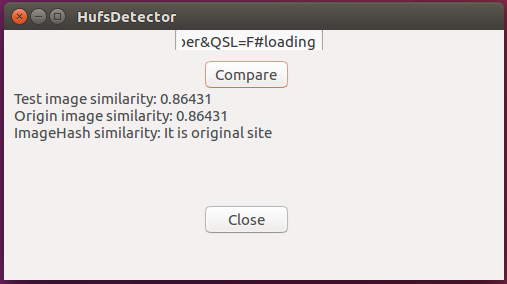
[그림 4-23]



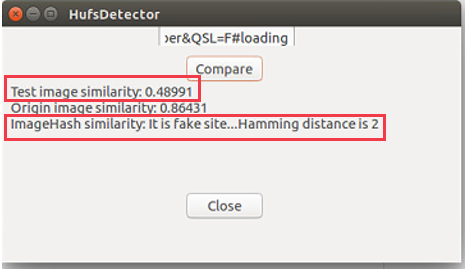
[그림 4-24]



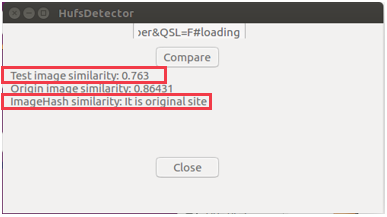
[그림 4-25]



[그림 4-26]



[그림 4-27]



[그림 4-28]

[그림 4-20]은 오리지널 웹사이트 이미지이고, [그림 4-21]과 [그림 4-22]는 빨간색 박스로 표시한 부분만 임의로 변경한 위조 웹사이트 이미지이다. [그림 4-23], [그림 4-24], [그림 4-25]는 [그림 4-20]. [그림 4-21]. [그림 4-22]의 유사도 결과이며 [그림 4-26], [그림 4-27], [그림 4-28]은 프로그램을 실행시켰을 때의 화면이다. 프로그램을 실행시켰을 때 오리지널 이미지의 경우 [그림 4-23]에서처럼 유사도가 서버에 저장된 오리지널 이미지의 유사도와 같으며 이미지 해시의 결과도 해밍 거리가 0이 나오기 때문에 [그림 4-26]과 같이 “It is original site”라는 종합적 결과가 User에게 보여지게 된다. 반면, 위조 이미지인 [그림 4-21]로 프로그램을 실행시켰을 경우 유사도가 서버에 저장된 오리지널 이미지의 유사도보다 낮게 나오며 이미지 해시 결과 또한 해밍 거리가 0이 아니기 때문에 종합적으로 “It is fake site”라는 결과가 User에게 보여지게 된다. 그런데, 또 다른 위조 이미지인 [그림 4-22]로 프로그램을 실행시켰을 경우에는 원래 오리지널 이미지가 있는 부분을 위조 이미지로 덮어씌운 것이기 때문에 오리지널 이미지와 구별을 하지 못해서 이미지 해시의 결과에서 해밍 거리가 0이 나오게 된다. 이 부분에서 이미지 해시의 탐지 한계를 보여준다.

1. 결론

인터넷을 이용한 업무와 금융서비스의 보편화로 인하여, 사용자가 늘어남에 따라 정부나 금융기관 사이트를 모방한 위•변조 사이트가 증가하고 있다. 또한 일반적인 사용자들은 위•변조 사이트를 쉽게 구분해 내기 어렵기 때문에 피해 또한 증가하고 있다.

본 논문에서는 웹사이트의 위•변조를 탐지 하기 위하여 URL과 이미지 학습을 이용하여 위•변조를 탐지하는 서비스를 제안 하였다. 제안하는 서비스는 사용자가 위•변조된 웹사이트에 접속하거나, 의심이 드는 웹사이트에 접속을 했을 때, 사용자는 해당 프로그램에 접속한 웹사이트나 접속할 웹사이트의 URL을 프로그램에 입력하여 URL 비교를 하여 URL이 오리지널 웹사이트와 일치하는지 여부를 먼저 확인 한다. URL 비교는 서버에서 크롤링 과정에서 저장된 해당 오리지널 사이트의 URL을 텍스트에 저장하여 비교 하려는 URL과 일치하는지 문자열 비교를 통해 비교한다. 이미지 비교 단계에서는 이미지 크롤링으로 오리지널 사이트의 이미지를 저장하여 데이터 셋을 만든 후 GoogleNet의 모델인 inception V3를 이용하여 이미지 학습을 하였다. URL 비교단계에서 URL이 일치하면 그 해당 URL의 웹사이트 이미지를 크롤링하여 저장한 후 서버로 전송한다. 서버에서는 사용자가 보낸 웹사이트 이미지를 기존 오리지널 이미지와 비교를 한다. 또한 dHash 알고리즘 사용하여 이미지 유사도에 정확성을 더 부여한다. 이렇게 이미지 유사도 결과를 종합하여, 오리지널 웹사이트 이미지와 사용자가 보낸 이미지와 유사도가 동일하면 오리지널 사이트로 판별하고, 아니면 가짜사이트라고 사용자에게 알려준다. 이러한 과정으로 웹사이트 위•변조에 대한 탐지를 한다.

향후 연구과제로 제시 하는 것은 본 논문의 이미지 전처리 단계에서 많은 양의 이미지를 저장하기 때문에 때문에 웹사이트의 갱신 주기가 짧을 경우, 그 갱신된 이미지의 실시간 저장이 어려울 수 있다는 것이다. 이러한 문제는 전처리 과정을 나누어 멀티프로세서로 수행하면 해결할 수 있을 것으로 보인다.

또한 다수의 유저가 동시에 서버에 요청을 보냈을 때 처리가 어려운 부분이 있는데, 이미지 학습 특성상 메모리의 사용량이 높기 때문에 메모리 분할과 같은 방법으로 메모리 문제를 해결하고, 멀티스레드로 구현하여 다수의 유저의 요청을 받을 수 있게 하도록 연구가 필요할 것이다.

1. 개발 일정



1. 참고 문헌

[1]-<https://m.post.naver.com/viewer/postView.nhn?volumeNo=16585222&memberNo=25598567>

[2],[3] - <https://www.ahnlab.com/kr/site/securityinfo/secunews/secuNewsView.do?seq=23390>

[4]-https://www.earticle.net/Article/A263848 – 이미지를 이용한 웹사이트 위•변조 탐지

[5]-PENG YANG, GUANGZHEN ZHAO , AND PENG ZENG, “Phishing Website Detection Based on Multidimensional Features Driven by Deep Learning”, **Published in:**[IEEE Access](https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=6287639) ( Volume: 7 ), 11 January 2019

[6]-F.C. Dalgic1, A.S. Bozkir 2\* and M. Aydos 3, “Phish-IRIS: A New Approach for Vision Based Brand Prediction of Phishing Web Pages via Compact Visual Descriptors”, **Published in:**[2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)](https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=8543123), 19-21 Oct. 2018

[7]-http://solarisailab.com/archives/2351

[8]-<https://yamerong.tistory.com/40>

[9]-https://jsideas.net/Inception\_v3\_transfer\_learning/

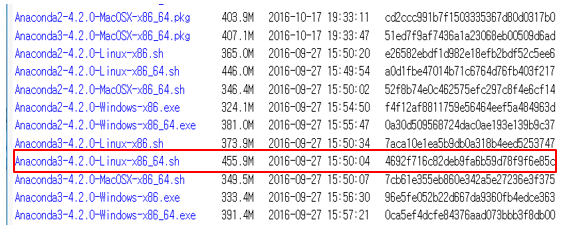
[10]-https://www.pyimagesearch.com/2017/11/27/image-hashing-opencv-python/

**GitHub URL-https://github.com/yjinsol/grad\_project.git**

1. 부 록

**1. Anaconda(아나콘다) 설치**

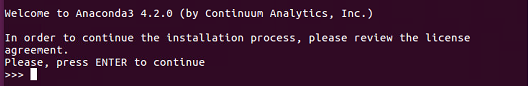
* 파이썬 2.x가 우분투에 설치되어 있을 경우, 아나콘다 2를 설치
* 파이썬 3.x가 설치되어 있을 경우 아나콘다 3을 설치합니다. (본 설치는 파이썬 3.5 설치 기준으로 하였으며, 아나콘다3 4.2를 설치하였습니다.)
* 아나콘다3 4.2는 텐서플로우와 호환성이 좋고, 파이썬 3.5까지 지원합니다.
* <https://repo.continuum.io/archive/> <- 아나콘다 저장소 링크에서 아나콘다를 다운받습니다.



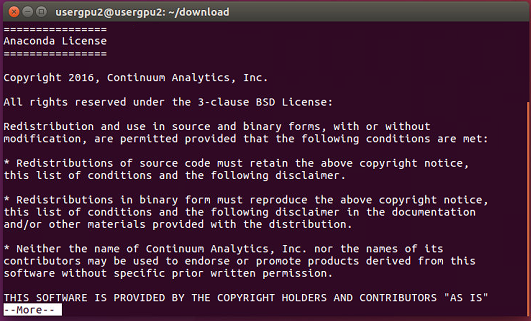
* 빨간표시를 한 Anaconda3-4.2.0-Linux-x86\_64.sh를 다운받습니다.



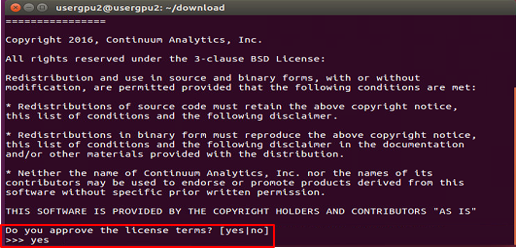
* 아나콘다 파일을 다운받은 경로로 들어가서 위와 같이 입력하고 설치를 진행합니다.



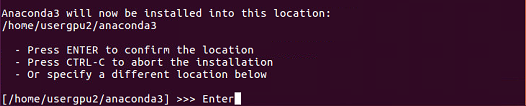
* 위와 같은 화면이 나오는데 Enter를 누릅니다.



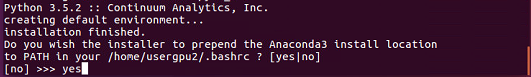
* 라이선스 관련 내용이 나오는데 시간상 Ctrl+c를 눌러서 skip합니다.



* yes를 입력해서 라이선스 조항에 대해 동의해줍니다.



* 동의 후 설치 경로에 설정 부분 입니다. Enter를 입력해서 해당 경로로 설정합니다.



* 환경설정이 완료 후, yes를 입력하여 해당 경로에 아나콘다를 설치합니다.



* 설치 완료 후 경로 설정 확인을 위해 위와 같이 입력합니다.



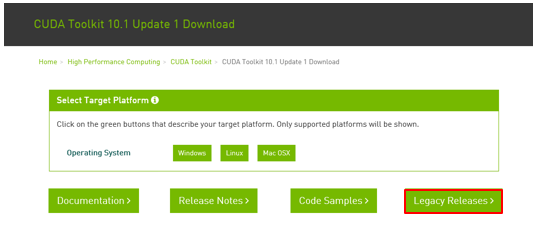
* 맨 아래로 내려서 #added ~ 빨간 표시로 된 경로가 입력이 되어있으면 경로 설정이 완료 된 것입니다.
* 경로가 작성이 안되어 있을 경우export PATH=”/home/사용자명/anaconda3/bin:$PATH”을 입력해줍니다.
* Esc + wq를 누르고 저장하고 나옵니다.

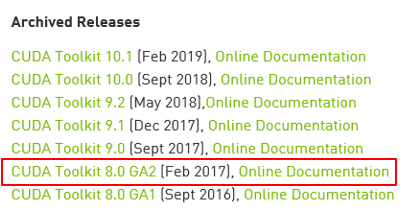


* 터미널 창에 python을 입력해서 위와 같은 화면이 나오면 정상적으로 설치가 완료가 되었습니다.

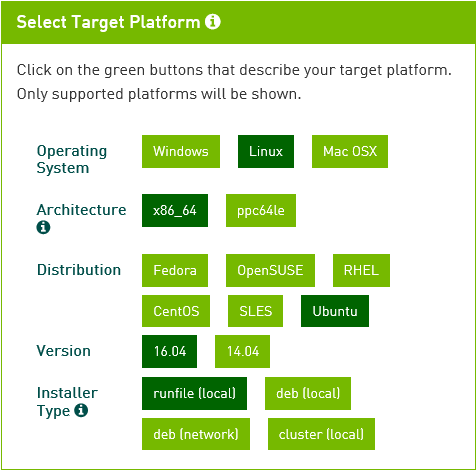
**2. CUDA Toolkit 설치**

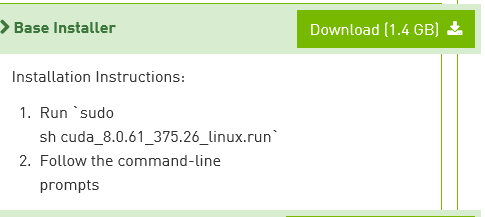
* <https://developer.nvidia.com/cuda-downloads> <- 경로로 들어가서





* Legacy Releases 클릭 후, 빨간 표시 8.0 GA2를 클릭합니다.

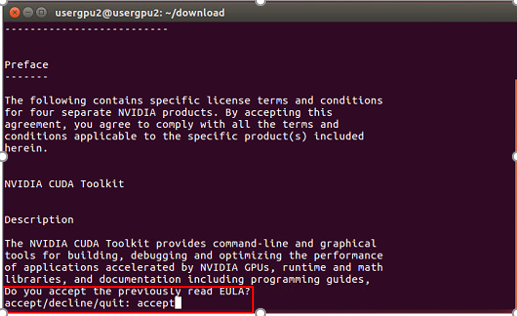




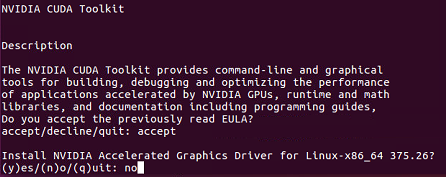
* 자신이 사용하는 컴퓨터 사양에 맞게 클릭하고 base installer를 다운받아줍니다. (이유는 자세하게 모르겠으나, 기존에 컴퓨터에 cuda 9.0이 설치되어 있었는데, tensorflow와 호환이 되지 않는건지, 우분투16.04의 파이썬 3.5버전과 맞지 않는건지 gpu로 학습이 진행되지 않아서 cuda 8.0을 사용하였습니다.)



* 위와 같이 입력하여 CUDA설치를 진행합니다. (다른 버전을 다운받으셨다면 그 버전 이름에 맞게 패키지명을 수정하시면 됩니다.)



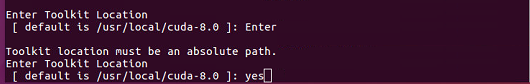
* 아나콘다와 마찬가지로 라이선스 관련 내용이 나옵니다. Ctrl + c를 눌러서 Skip 후에 accept를 입력해서 동의 해줍니다.



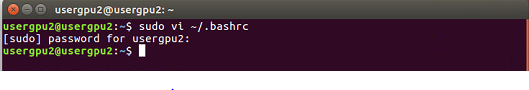
* 그래픽카드 설치 관련 내용이 나오는데, 사용하는 현재 환경에서는 그래픽 드라이버가 설치되어 있으므로 no를 입력합니다. (그래픽 드라이버가 설치 안되어 있다고 yes입력하지말고, 그래픽카드는 CUDA 설치 전 따로 설치합니다.



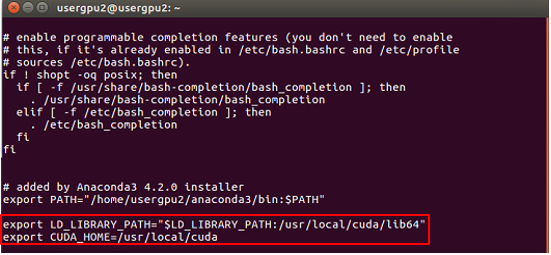
* yes를 입력해서 툴킷을 설치합니다.



* 툴킷 경로를 설정해주기위해 Enter를 입력하여 해당 경로로 설정하고, yes를 입력 후 절대 경로로 설정하여 설치를 진행합니다.
* 다음에 Install the CUDA 8.0 Samples? 내용이 나오는데 no를 입력해줍니다. (y)es/(n)o/(q)uit: no



* 설치 완료 후 위와 같이 입력하여 아나콘다와 마찬가지로, 맨 밑에 아래와 같이 작성해줍니다.

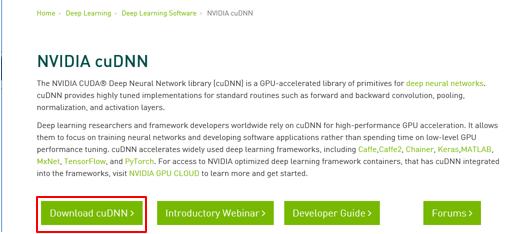


* 저장 후 아래와 같이 꼭 입력해줍니다.



**3. CUDNN 설치**

* <https://developer.nvidia.com/cudnn> <- 해당 경로로 들어가서 CUDNN v5를 설치합니다. (설치를 위해서는 NVIDIA가입을 해야합니다. )

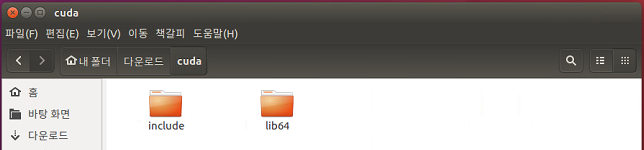




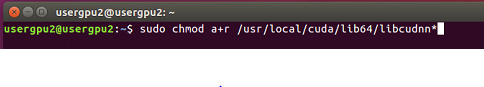




* 위 그림 빨간 박스 표시된 곳을 따라서 다운로드 후 다운 받은 경로에 가서 압축을 풀면 CUDA폴더가 생성됩니다.
* 터미널을 실행하여 sudo nautilus 입력해서 관리자 권한으로 탐색기 창을 띄웁니다.



* CUDA안에 있는 폴더에서 include에있는 파일을 usr/local/cuda/include안에 붙여 넣고, 다음 lib64 안에 있는 모든 파일을 usr/local/cuda/lib64안에 붙여 넣습니다.



* 새로운 터미널창을 열어서 위와 같이 입력해서 권한을 부여해 줍니다. 이렇게 하면 아나콘다, CUDA, CUDNN의 설정이 완료되었습니다.

\* 설치된 버전 확인 (터미널 창에 입력하면 됩니다.)

* CUDA 버전확인 -> nvcc --version
* CUDNN 버전확인 -> cat /usr/include/cudnn.h | grep CUDNN\_MAJOR -A 2 python -c 'import tensorflow as tf; print(tf.\_\_version\_\_)'

**4, TensorFlow 설치**



* pip install –ignore-installed –upgrade <https://storage.googleapis.com/tensorflow/linux/gpu/tensorflow_gpu-1.2.1-cp35-cp35m-linux_x86_64.whl> 을 입력해줍니다. (파이썬 버전에 맞게 3.6이라면 cp36-cp36m으로 수정해줍니다. 기본적으로 텐서플로우가 설치 되어있으면 설치 할 필요 없습니다.)
* 새로운 터미널 창을 열어서 python을 입력 후 아래와 같이 입력합니다.

>>> import tensorflow as tf

>>> hello = tf.constant('Hello, TensorFlow!')

>>> sess = tf.Session()

>>> print(sess.run(hello))



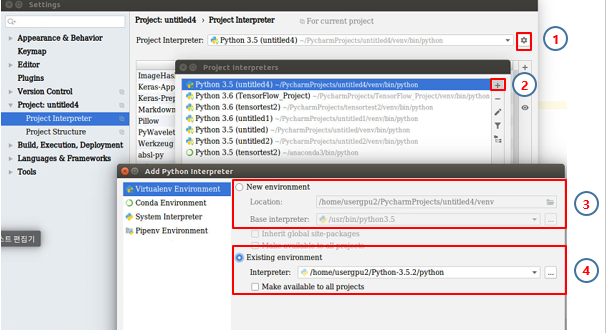
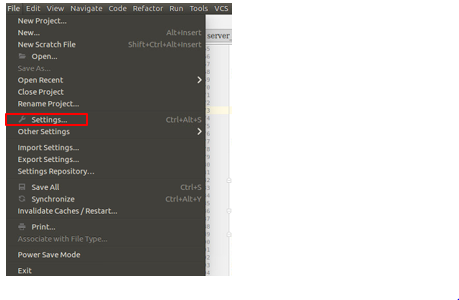
* 위와 같이 결과가 나오면 정상적으로 설치가 완료된 것입니다.

**5. Pycharm설치**

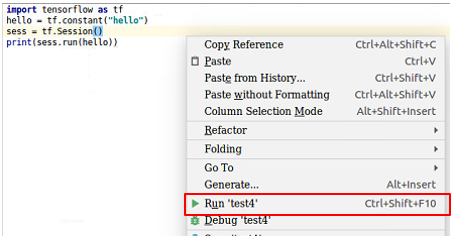
* <https://www.jetbrains.com/pycharm/download/#section=linux> <- 경로에 들어가서 pycharm을 다운받습니다. (Community로 다운로드하셔도 됩니다. Professional은 라이센스 등록도 해야 해서 번거롭습니다.)
* 다운로드 후 압축을 풀고 아래와 같이 터미널 창에 입력합니다.



* 그 다음 bash pycharm.sh을 입력하여 설치를 진행해주고 파이참을 실행하여 프로젝트를 생성합니다.



* 위 그림에서 file -> setting을 들어갑니다.
* 컴파일 환경 설정을 위해 1번 -> 2번 클릭 후, 아나콘다 환경에서 텐서플로우를 실행하실 분들은 3번에 경로를 anaconda 설치 경로로 설정해주시고, 기존에 텐서플로우가 설치되어 있거나 가상 환경에서 돌리지 않으실 분들은 Existing environment를 클릭해서 컴퓨터에 설치되어있는 파이썬 버전에 맞는 경로로 위와 같이 설정해주시면 됩니다.



* 텐서플로우 설치 확인과 똑같이 파이참에도 위와 같이 작성해서 실행해서 결과가 똑같이 나오면 정상적으로 설치가 완료된 것을 확인 하실 수 있습니다.

**\* 설치 유의 사항**

* CUDA와 CUDNN, Tensorflow가 기존에 설치되어있음에도 불구하고, GPU로 안돌아가거나 환경이 안잡힐 수 있습니다. 우분투 버전과 파이썬 버전과 각각의 툴킷을 버전을 잘 확인하셔야 합니다.
* 무작정 최신버전이 좋은건 아니고 호환이 되어야 하기 때문 입니다. 무리하게 사용하고있는 파이썬 환경을 업그레이드 하지 말고, 사용하고있는 파이썬 버전에서 최적화된 cuda와 tensorflow를 설치하셔야 합니다. (잘못하면 Ubuntu 환경 자체가 다운 될 수 있습니다.

**[출처 및 참고]**

<http://sldofvge12.blogspot.com/2018/04/ubuntu-tensorflow-gpu-anaconda-pycharm.html>