**졸업논문**

**URL과 이미지 비교를 이용한 웹 사이트**

**위•변조 탐지 및 알림 서비스**

**Ver. 1.3**

**2019. 05. 19**

**한국외국어대학교 정보통신공학과**

**201403586 허성윤**

**201602119 유진솔**

지 도 교 수 : 정성호

**문서 정보**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 구 분 | 소 속 | 성 명 | 날 짜 | 서 명 |
| 작성자 | 한국외국어대학교 | 유진솔 | 2019. 05. 19 |  |
| 한국외국어대학교 | 허성윤 | 2019. 05. 19 |  |
| 검토자 | 한국외국어대학교 | 유진솔 | 2019. 05. 19 |  |
| 한국외국어대학교 | 허성윤 | 2019. 05. 19 |  |
| 승인자 | 한국외국어대학교 | 정성호 | 2019. 05. 19 |  |

**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 그에 대한 알림 서비스를 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 | | | | |
| 1.3 | 허성윤 | 2019. 05. 19 | 본론 작성(1) | |  | |
| 유진솔 |
| 검토자 | 허성윤, 유진솔 | | | | |

목 차

목차

[목 차 5](#_Toc9192514)

[1. 서론 9](#_Toc9192515)

[1.1 서비스 정의 9](#_Toc9192516)

[1.2 서비스 기획 배경 및 목적 9](#_Toc9192517)

[1.2.1 웹사이트 위•변조 공격 – 피싱(Phishing)과 파밍(Pharming) 9](#_Toc9192518)

[1.2.2 웹사이트 위•변조 공격 – hosts 파일 및 hosts.ics 파일 변조 10](#_Toc9192521)

[2. 관련 연구 조사 및 분석 11](#_Toc9192525)

[2.1 Feature Matching 기반 탐지 기법 11](#_Toc9192526)

[2.2 URL 기반 탐지 기법 11](#_Toc9192529)

[2.3 시각적 추출 기법(Visual Descriptor)과 SVM을 이용한 탐지 기법 12](#_Toc9192536)

[3. 주제 제안 13](#_Toc9192537)

[3.1 시나리오 13](#_Toc9192538)

[3.1.1 URL 일치 여부 확인 13](#_Toc9192539)

[3.1.2 웹 크롤링을 이용한 웹사이트 이미지 저장 13](#_Toc9192540)

[3.1.3 Inception-v3를 이용한 이미지 유사도 검출 14](#_Toc9192541)

[3.1.4 dHash를 이용한 이미지 유사도 검출 14](#_Toc9192542)

[3.2 사용 기술 설명 14](#_Toc9192543)

[3.2.1 TensorFlow 14](#_Toc9192544)

[3.2.2 Transfer Learning 15](#_Toc9192545)

[3.2.3 dHash Algorithm 15](#_Toc9192546)

[4. 시스템 구성 16](#_Toc9192547)

[4.1 시스템 구성도 16](#_Toc9192548)

[4.2 시스템 동작 순서 17](#_Toc9192549)

[4.2.1 URL 비교 17](#_Toc9192550)

[4.2.2 이미지 전처리 18](#_Toc9192552)

[4.2.3 이미지 유사도 비교 19](#_Toc9192555)

[5. 개발 일정 22](#_Toc9192556)

[6. 참고 문헌 23](#_Toc9192557)

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.





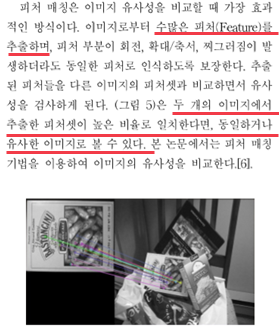
[그림 1-2]

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

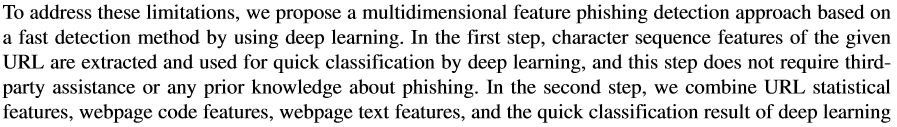


[그림 2-1]

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]

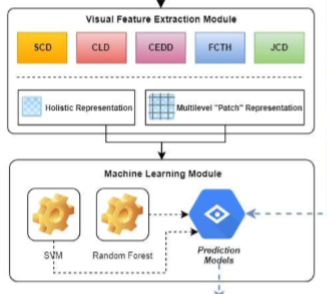
위 논문은 URL을 기반으로 위•변조 웹 사이트를 탐지한 연구이다. 이 연구에서는 URL 시퀀스

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



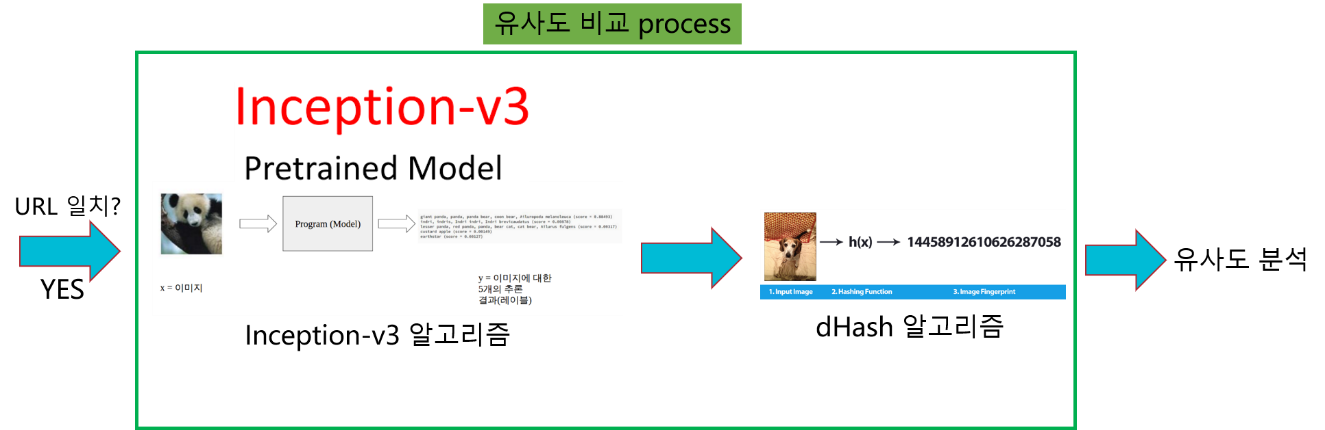


**[그림 2-3]**

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 은행에도 같은 기술이 적용될 수 있음을 보인다.

3.1.1 URL 일치 여부 확인

* 1. 사용자의 URL을 입력 받는다..
  2. Python selenium으로 캡쳐한 웹 사이트 화면의 이미지 이름을 해당 URL로 저장한다.
  3. 이미지들의 이름, 즉 URL 리스트를 텍스트 파일로 생성한다.
  4. 사용자가 입력한 URL이 텍스트 파일에 있는 URL과 일치하는지 비교한다.
  5. 일치하지 않으면 변조사이트로 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 BeautifulSoup으로 해당 URL의 html에서 다른 링크로 갈 수 있는 URL만을 추출한다.
3. 추출한 URL들 중 https로 시작하는 유효한 URL만 다시 추출한다.
4. 유효한 URL에 접속하여 웹사이트 이미지를 캡쳐하고 해당 이미지의 URL이름으로 저장한다.
5. 이 때, URL 형식 그대로 저장할 수 없기 때문에 저장이 가능하도록 특정 특수문자를 바꾸어 저장한다.
6. 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 사이트 별로 폴더로 분류하고 이 폴더를 데이터 셋으로 사용한다.
3. Inception V3를 사용하여 이미지 학습을 한다.
4. 오리지널 사이트의 데이터 셋을 input으로 넣어주고 이 이미지들을 (n, 299, 299 3)의 배열로 변환한다.
5. Convolution Layer와 Pooling Layer과정을 거쳐서 (n, 2048)의 배열로 변환한다.
6. Fully connected로 m개의 class로 분류한다.
7. Transfer Leaning을 위한 기존의 이미지 학습 모델의 오픈소스인 retrain.py를 이용하여 이미지 학습을 하여 Label을 생성한다.
8. 학습된 결과로 나온 Output Label과 사용자가 비교할 이미지와 비교하여 유사도를 검출한다.

3.1.4 dHash를 이용한 이미지 유사도 검출

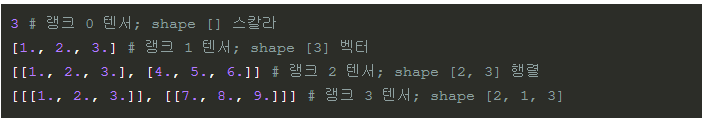
* 1. 오리지널 사이트의 이미지를 데이터베이스에 미리 저장해 둔다.

1. 웹 사이트에 접속하여 처음 보이는 홈 화면을 web application test framework의 일종인 selenium을 이용하여 캡쳐 후 로컬 파일에 자동 저장한다. (추가적인 비교를 위해 URL도 함께 저장한다.)
2. 웹 페이지의 유동적인 부분들로 인해 몇 분 간격으로 바뀌는 것까지 최대한 ‘같은 페이지의 많은 이미지’를 저장하기 위해 일정 시간 간격으로 캡쳐 하도록 한다.
   1. 오리지널 사이트의 이미지와 사용자가 접속한 사이트의 이미지를 비교하여 유사도를 검출한다.
3. Python에서 제공하는 libarary인 ImageHash에서 처리 속도가 빠르고 이미지 비교에 적합한 dHash 알고리즘을 사용한다.
4. 결과값으로 두 이미지의 차이 정도를 나타내는 해밍 거리 값을 얻는다.
   1. 검출된 유사도 정보로 사용자가 접속하고있는 사이트에 대한 유사도 정보를 제공한다.
5. 결과 해밍 거리 값을 가지고 오리지날과 유사한 정도를 백분율로 환산하여 100%라면 같은 이미지, 95%~99%라면 유사한 이미지, 그 이하라면 다른 이미지라고 판단한다.
6. 이 정보를 가지고 이 웹사이트의 위,변조 가능성을 경고하는 알림을 띄워준다.

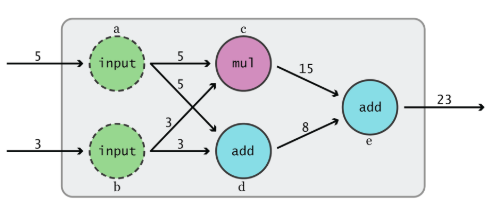
3.2 사용 기술 설명

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]**

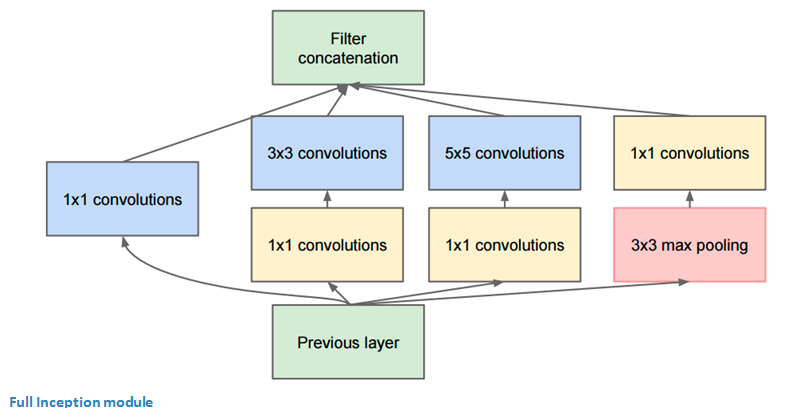


**[그림 3-3]**

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

3.2.2 Transfer Learning

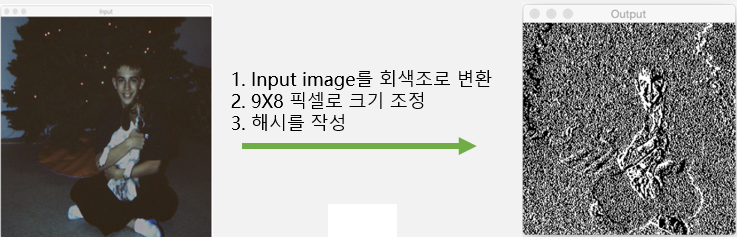
기존의 만들어진 모델을 사용하여 새로운 모델을 만들 시 학습을 빠르게 하며, 예측력을 높이는 방법이다.

**[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

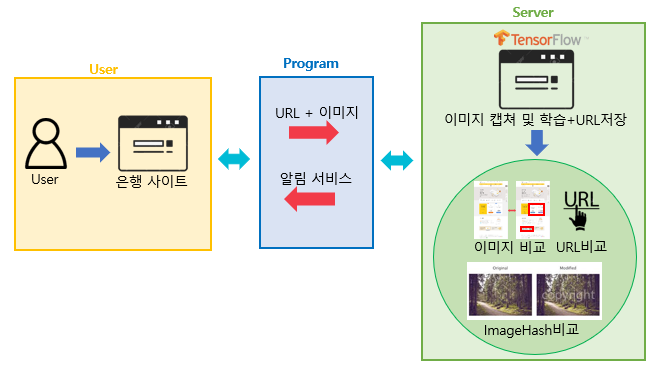


**[그림 3-5]**

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

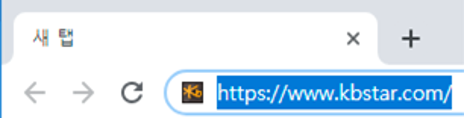


**[그림 4-1]**

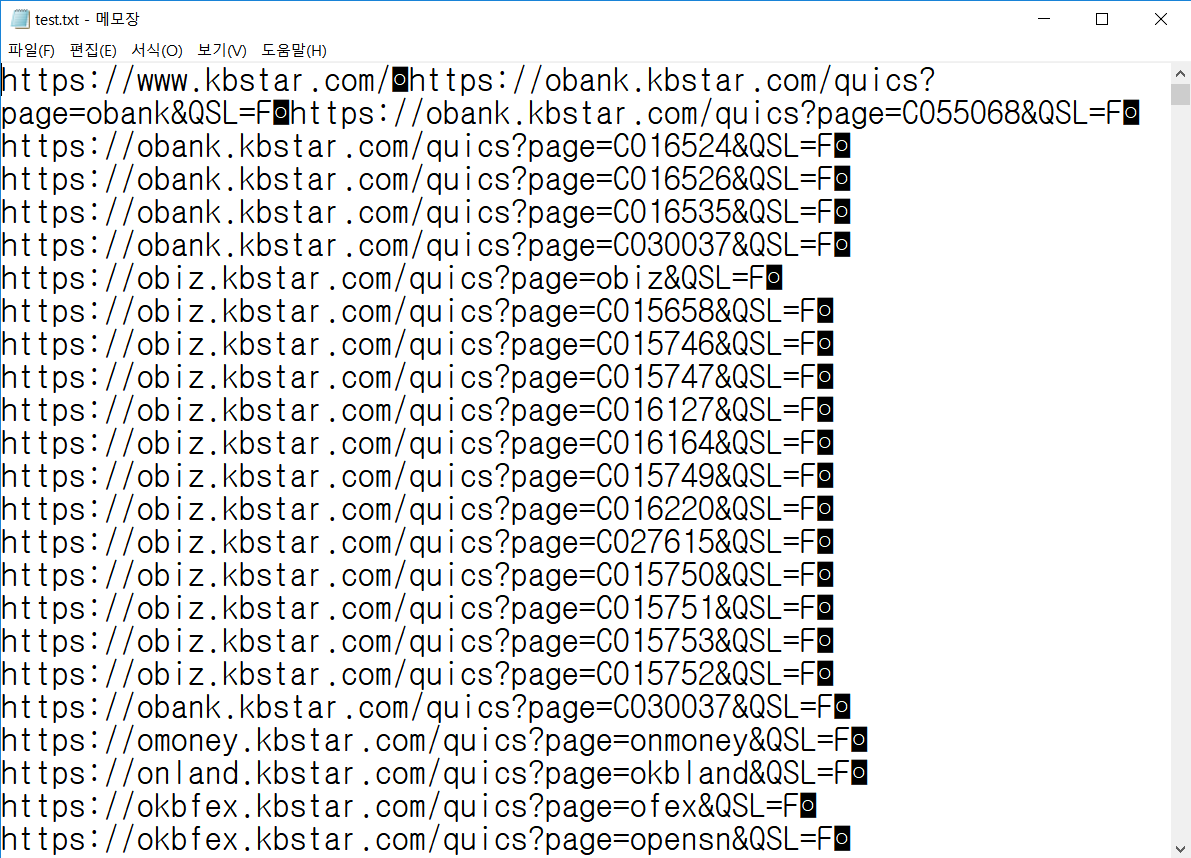
User가 은행 웹사이트에 접속하면 실행 프로그램에서 User가 접속해 있는 URL과 캡쳐된 이미지를 받아 서버로 전송한다. 서버에서는 프로그램으로부터 받은 URL이 이전에 미리 저장해둔 Original URL list에 있는지 확인 후 없다면 프로그램을 통해 Fake site라고 User에게 알림을 보낸다. 만약, Original URL list에 있는 URL이라면 이미 학습시켜 둔 해당 은행 웹사이트 이미지와 프로그램으로부터 받은 이미지를 비교하여 유사도를 검출한다. 또한, 정확성을 높이기 위하여 ImageHash로 한번 더 비교하여 종합적으로 결과를 내어 Original site인지 Fake site인지를 판별 후 프로그램을 통해 User에게 알림을 주게 된다.

4.2 시스템 동작 순서

4.2.1 URL 비교



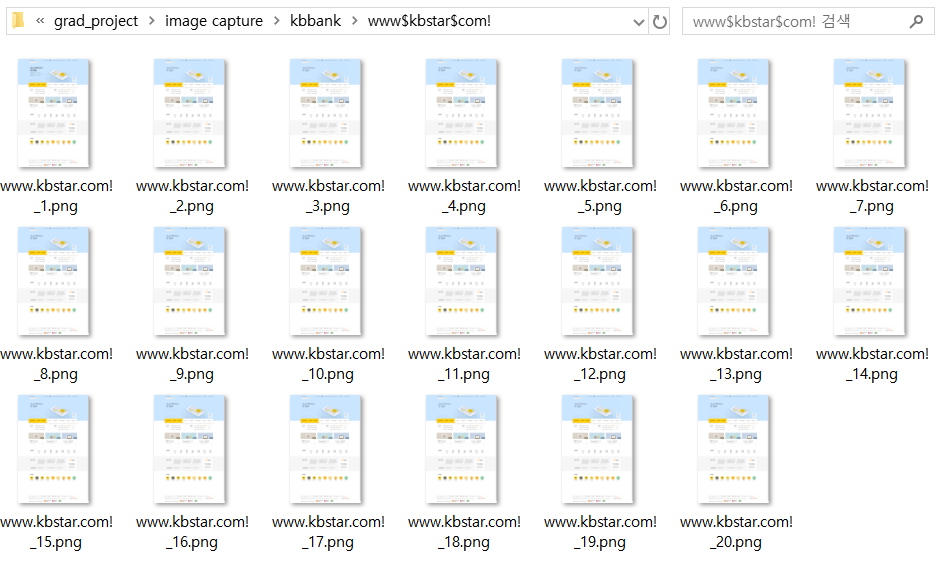
**[그림 4-2]**



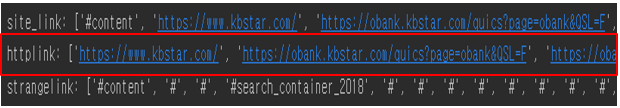
**[그림 4-3]**

[그림 4-2]와 같이 키보드 입력을 핸들링하는 Python pynput keyboard를 이용하여 User가 URL을 직접 입력 후 엔터를 눌렀을 경우나 URL을 드래그해서 Ctrl+C를 눌렀을 경우에 접속한 웹페이지의 URL을 받아올 수 있도록 하였다. [그림 4-3]은 국민은행 웹사이트를 크롤링하면서 동시에 저장한 Original URL list이다. User로부터 받아온 URL을 이 Original URL list에 저장된 URL들과 비교하여 일치하는 것이 있다면 ‘This URL is matching’이라는 메시지를 서버에 전송하고 서버에서는 이미지 비교를 시작한다. 만약, 일치하는 URL이 없다면 ‘There is no match’라는 메시지를 서버에 전송한다. 이 때, 서버에서는 이미지 비교를 하지 않고 User에게 바로 Fake site라는 알림을 줄 수 있도록 프로그램을 동작시킨다.

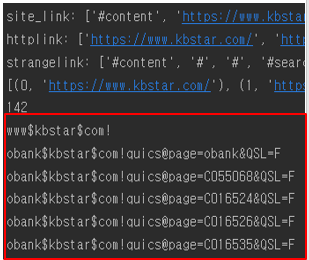
4.2.2 이미지 전처리

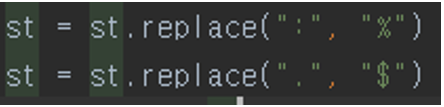


**[그림 4-4]**

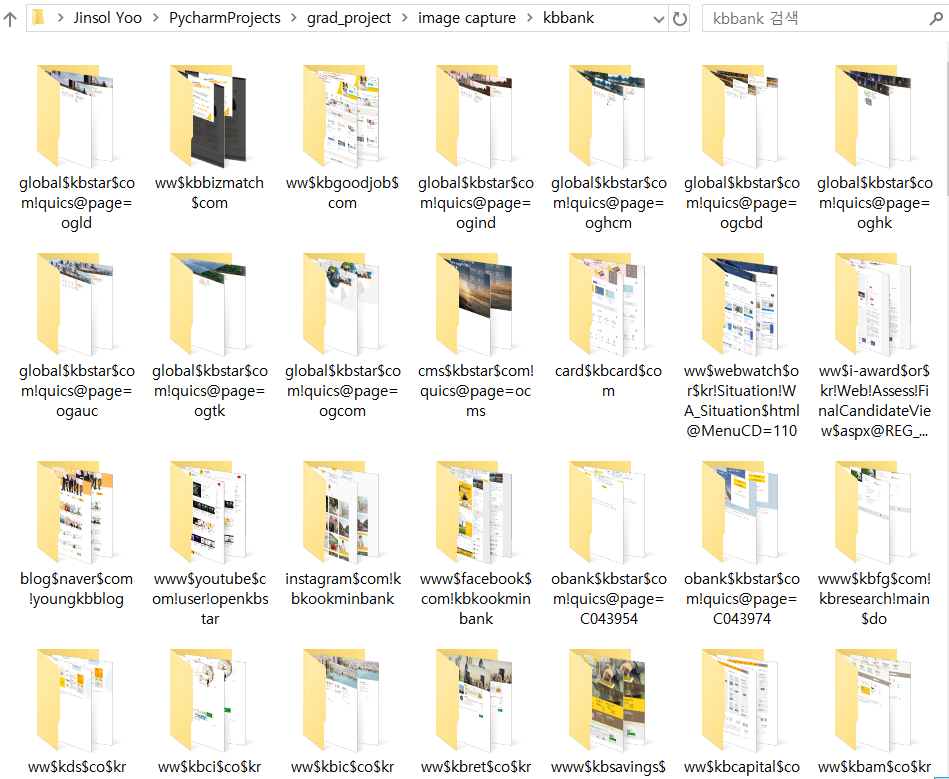


**[그림 4-5]**





**[그림 4-6]**



**[그림 4-7]**

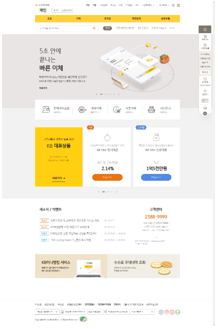
하나의 은행 웹사이트에서 접속할 수 있는 여러 웹페이지들의 이미지까지 최대한 많은 수의 이미지를 캡쳐하기 위해 웹사이트 html에서 현재 링크로부터 다른 링크로 넘어갈 수 있는 URL부분, 즉 <a href>태그 부분의 URL만을 추출한다. [그림 4-5]와 같이 추출된 URL들 중 https로 시작하지 않는 URL들은 제거하고 유효한 URL들만 다시 추출한다. 유효한 URL로 접속하여 캡쳐되는 수많은 이미지들을 구별하기 위해서 [그림 4-4]와 같이 추출한 URL을 이미지 파일 이름으로 저장하였다. 사이트에서 새로고침 할 때마다 다른 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 141개 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 유사도 비교

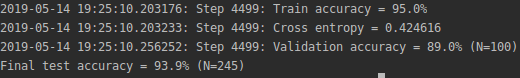


**[그림 4-8]**

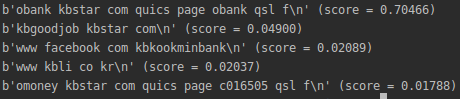
이미지 비교 테스트를 위한 가짜 은행 사이트 이미지를 구하기 쉽지 않았기 때문에 직접 위조 사이트를 만들었다. 본 논문에서는 실제 국민은행 사이트의 html에서 이미지 부분을 우리은행 이미지로 변경하였고, 도메인 주소도 국민은행의 도메인과 비슷하게 하여 제작하였다.

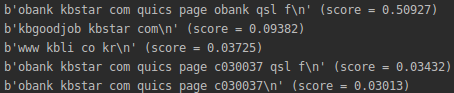
**[그림 4-9] [그림 4-10]**



**[그림 4-11]**

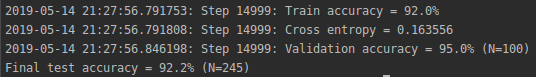


**[그림 4-12]**

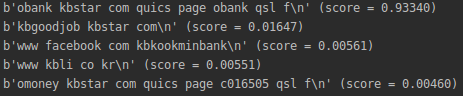


**[그림 4-13]**

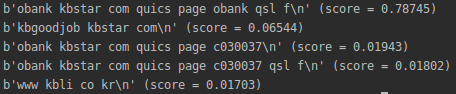
[그림 4-9]은 기존 사이트, [그림 4-10]은 위에서 제작한 위조 사이트이다. 이 이미지들을 가지고 학습 횟수에 따른 유사도를 비교하였다. 앞서 크롤링을 이용하여 저장한 웹사이트 이미지들을 Dataset으로 하여 Inception-v3 모델을 사용해서 이미지 학습을 진행하였다. [그림 4-11]은 4500번 학습시킨 결과, 학습 정확도 93.9%를 보여주고 있다. [그림 4-12]와 [그림 4-13]은 4500번 학습 시킨 후 [그림 4-9]와 [그림 4-10]으로 테스트한 결과이다. [그림 4-12]는 기존 사이트의 유사도 70.5%를, [그림 4-13]은 위조 사이트의 유사도 50.9%를 나타낸다.



**[그림 4-14]**

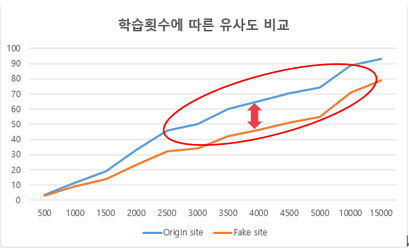


**[그림 4-15]**



**[그림 4-16]**

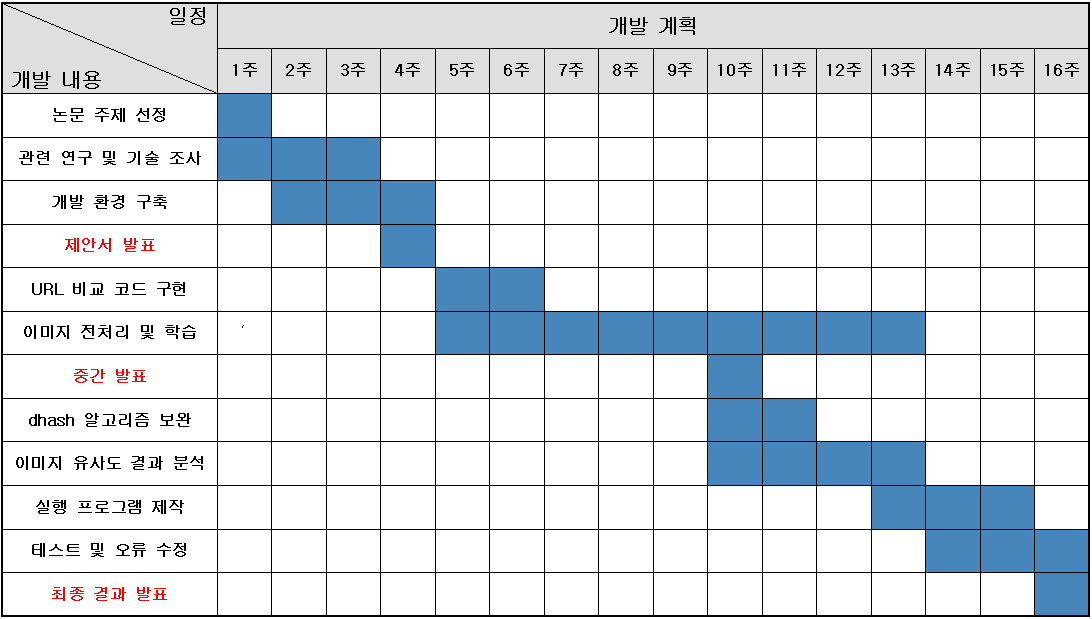
[그림 4-14], [그림 4-15], [그림 4-16]은 학습 횟수를 15000번으로 하고 위에서 사용한 이미지와 같은 테스트 이미지를 사용하여 테스트했을 때의 결과이다. [그림 4-14]는 학습 정확도 92.2%를 보여주고 있고, [그림 4-15]는 기존 사이트의 유사도 93.3%, [그림 4-16]은 위조 사이트의 유사도를 나타낸다.



**[그림 4-17]**

[그림 4-17]은 이런 식으로 학습 횟수를 증가시키면서 학습시킨 후 [그림 4-9]와 [그림 4-10]으로 테스트한 결과를 그래프로 나타낸 것이다. 3500번 학습시킨 구간 이후부터 10000번 학습시킨 구간까지 [그림 4-9]와 [그림 4-10]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.

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**