Convolutional Neural Network을 활용한 패킷 페이로드 기반 네트워크 트래픽 분류

김주봉, 임현교, 허주성, 한연희*
한국기술교육대학교 컴퓨터공학부
e-mail:{rlawnqhd, glenn89, chil1207, yhhan}@koreatech.ac.kr

Packet Payload-based Network Traffic Classification using Convolutional Neural Network

Ju-Bong Kim, Hyun-Kyo Lim, Joo-Seong Heo, Youn-Hee Han*
School of Computer Science and Engineering
Korea University of Technology and Education

요 약

네트워크 트래픽 데이터를 정제하여, Convolutional Neural Network Model 훈련에 적합한 데이터 세트로 변환하는데, 그 방법은 패킷 단위의 트래픽 데이터를 이미지 형태로 만드는 것이다. 완성된 데이터 세트를 훈련데이터로 하여 Convolutional Neural Network Model에 훈련하고, 훈련데이터의 이미지 크기를 변환해가며 훈련시킨 결과에 대해 비교 분석 및 평가를 진행한다.

1. 서론

최근 IoT시대 네트워크에서는 다양한 형태의 응용 및 서비스가 운영되고 있고, 이러한 다양한 네트워크에의 특성상 기존의 방식과 같은 수동적인 방식은 지금까지와는 차원이 다른 훨씬 더 크고 다양한 규모의 트래픽 데이터가 발생하고 있다. 따라서 사람이 해야 할 데이터 분석과 관리 범위가 상당히 넓어져야 한다는 것을 의미하며 [1], 이에 따라 경량화와 자동화가 된 망의 구축이 필요하고 [2], 더불어 최근 각광받는 딥러닝 기법의 활용도가 높아지고 있다.

딥러닝 관련 기법에는 대표적으로 Multilayer Neural Network Model (MNN) [3], Convolutional Neural Network Model (CNN) [3], Recurrent Neural Network Model (RNN) [3] 등이 있다. MNN은 약 3개에서 7개까지의 Hidden Layer로 구성된 기본적인 딥러닝 모델이며, CNN은 Local Receptive Fields과 Convolutional Layer, Pooling Layer 세 단계로 구성되어 있는 Neural Network로써, 이미지 분석에 대표적으로 쓰이고 있는 딥러닝 기법이다. RNN은, 임의의 순차 데이터(Sequenced Data)를 처리하기에 적합한 순환 신경망에 넣어 출력 데이터를 뽑아내는 데, 출력 데이터는 이전의 연산결과에 영향을 받는 것이 특징이다.

본 논문의 2장에서는 네트워크 트래픽 데이터의 정제과정을, 그리고 3장에서는 CNN 학습에 사용할 데이터 세트 변환 작업에 대해 설명한다. 4장은 실험에 사용한 CNN의 구조에 대해서 설명하고, 5장은 실험 및 평가를 진행한다.

2. Traffic Data Munging

본 논문에서는 Network Traffic Data로써, Broadband Communications Research Group [4]에서 제공받은 PCAP 파일을 사용한다. PCAP (Packet Capture)파일은 Wireshark, TCPdump등과 같은 프로그램을 이용하여 네트워크 패킷을 캡처하여 파일 형태로 저장한 것이 PCAP 파일이다. 본 실험에서는, PCAP 파일을 이미지 데이터세트로 가공하여 CNN을 통한 학습 및 실험과 평가를 시행하였다. 실험에 앞서, Application Layer의 Payload를 패킷 단위로 나눈 문자열들을 이미지 데이터세트로 Conversion하게 되는데, 이를 위해서는 PCAP 파일을 필터링하고 축약하는 데이터 정제시킬 필요가 있다.

원본 PCAP의 크기는 약 59GB이고, 769507개의 flow가 존재한다(그림 1). ‘packets_default.info’ 파일에는, 트래픽 데이터에 대한 labeling이 되어 있었다. labeling은 해당 트래픽 flow에 대한 응용타입, 프로토콜, 응용이름, 세부분류 등을 기재하는 것인데, PCAP 파일과 함께 제공 받았던 ‘packets_default.info’파일에는, flow에 대한 labeling이 자세하게 되어 있었고, 그 덕분에, 본 실험의 Ground Truth에 해당하는 정확한 label을 얻을 수 있었다. <표 1>의 ‘packets_default.info’의 데이터헤더는 labeling의 기준이 되며, flow 단위로 해당 flow의 5-tuple과 응용타입, 응용이름이 순차적으로 파일 내에 존재한다는 것을 알 수

* 교신 저자: 한연희 (한국기술교육대학교)
이 논문은 2016년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. NRF-2016R1D1A3B03933355)

있다.

Application	#Flows	#Megabytes
Edonkey	176 581	2 823.88
BitTorrent	62 845	2 621.37
FTP	876	3 089.06
DNS	6 600	1.74
NTP	27 786	4.03
RDP	132 907	13 218.47
NETBIOS	9 445	5.17
SSH	26 219	91.80
Browser HTTP	46 669	5 757.32
Browser RTMP	427	5 907.15
Unclassified	771 667	3 026.57

(그림 1) Broadband Communications Research Group PCAP Data Flow

데이터 정제과정에 들어가기에 앞서, flow수를 기준으로 가장 많은 label명 8종을 선별하였다. 상위 8종의 label명들은, Remote Desktop Protocol (RDP), Skype, SSH, Bittorrent, HTTP-Facebook, HTTP-Google, HTTP-Wikipedia, HTTP-Youtube 이고, 이 8종의 label이 붙은 flow들만을 PCAP파일에서 선택하고, 선택된 flow 내부 패킷의 Application Layer Payload만을 필터링하여 추출하였다. 추출된 Payload Data를 <표 1>의 두 번째 데이터헤더에 기준하여, 8개의 Application Layer Payload Data 파일을 만들었다. 그리고, 같은 HTTP (Hypertext Transfer Protocol)를 프로토콜로 사용하는, HTTP-Facebook, HTTP-Google, HTTP-Wikipedia, HTTP-Youtube, 4종의 HTTP트래픽을 Web이란 label로 통합하였고, 최종적으로 데이터세트의 label을, RDP, Skype, SSH, Bittorrent, Web, 총 5가지 종류로 결정하였다. 통합 과정을 거쳐 5개의 'Application Layer Payload Data' 파일이 완성 되었다. 데이터 파일은, flow마다 헤더 정보를 가지며, Payload는 패킷마다 '#'이란 구분자로 구분지었다. 즉, 각개의 파일은 같은 label을 가진 flow들의 집합이며, 패킷들의 Payload만을 가지고 있고, Payload는 패킷별로 구분되어진다. 또한 Payload는 문자열로써, 한 문자의 크기가 4bit이다. 아래의 (그림 2)는 완성된, 'Application Layer Payload Data', 파일들의 통계 정보를 나타낸다.

<표 1> info파일 데이터 헤더

파일명	데이터 헤더
packets_default.info	flow_id#start_time#end_time#local_ip#remote_ip#local_port#remote_port#transport_protocol#operating_system#process_name#labels#-#-#
Application_Layer_Payload_Data.info	flow_id#start_time#end_time#Payload#-#-#

3. Traffic Data Conversion

본 장에서는 CNN의 학습에 필요한 데이터세트를 제작하는 방법을 설명한다. 데이터세트는 500,000개의 'train data'와 'train label', 5,000개의 'validation data'와 'validation label', 10,000개의 'test data'와 'test label'을 갖는다. train, validation, test data는 앞선 2장의 Munging 작업을 거쳐 생성된 5개의 'Application Layer Payload Data'파일의 패킷 Payload를 이미지로 변환하는 작업을 거친다. 하나의 data 포맷은, $2^n \times 2^n$의 크기를 가진 2차원 Array인데, 패킷 하나의 Payload를 의미한다. 2차원 Array의 한 픽셀은 2^nbit의 크기를 가지고, 문자형 값을 부동소수 값으로 치환한 값을 갖는다. 즉, 2차원 Array Packet data는 수식 (1)과 같이, 그 크기를 표현 할 수 있다.

	rdp	skype	ssh	bittorrent	web
Total number of filtered flows	153,349	2,041	38,831	96,222	21,715
Total number of packets	6,875,730	3,015,153	17,910,988	207,306,103	47,136,213
Average number of packets in a flow	44	1477	461	2,154	2,483
Total packet size (GB)	131.3	12.0	321.7	2,170 (2.1TB)	698.4
Average size of all packets in a flow (MB)	0.9	6.0	8.5	23.7	39.9
Average packet size over all flows (KB)	20.0	4.2	18.8	11.3	15.9
Min packet size over all flows (B)	8.0	8.0	8.0	8.0	8.0
Max packet size over all flows (MB)	5.5	0.3	6.6	0.8	40.0

(그림 2) Statistics of Application Layer Payload Data Files

$$Packet\ data : 16 \times 16 = 256 \qquad (1)$$
$$256 \times 4 = 1024bit(128Byte)$$

0	3	0	0	0	0	3	4	0	2	15	0	8	0	6	8
0	0	0	1	0	3	14	11	7	0	2	6	0	8	0	0
1	0	0	0	0	6	5	12	9	0	12	5	3	4	6	6
4	12	8	8	2	14	9	0	9	5	1	7	1	3	6	7
7	11	11	11	1	12	6	4	2	1	2	8	0	3	11	1
8	6	1	11	6	3	15	13	13	9	7	5	9	4	5	8
10	13	9	6	10	10	5	14	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(그림 3) 16×16 패킷 이미지 데이터

(그림 3)은 완성된 패킷 이미지 데이터의 형태를 표현한 것이다. 위 패킷 이미지 데이터의 한 원소의 크기는 4bit이며, 부동소수 값으로 0.에서 15.사이의 값을 지닌다.

train, validation, test label은 총 다섯 가지의 응용에 대한 label을 가지고 있으므로, 길이가 5인 one-hot vector로써 표현을 하였다. one-hot vector label이란, 하나의 요

소만 1인 값을 지니는 벡터로써, 1인 값의 인덱스가 응용 이름을 가리키는 label로 정의 할 수 있다. <표 2>는 train, validation, test label이 one-hot vector로 어떻게 표현이 되어있는지를 나타낸다.

<표 2> 응용별 One-hot Vector Label

응용 이름	label
RDP	[1, 0, 0, 0, 0]
Skype	[0, 1, 0, 0, 0]
SSH	[0, 0, 1, 0, 0]
Bittorrent	[0, 0, 0, 1, 0]
Web	[0, 0, 0, 0, 1]

4. CNN 구조

본 장에서는 실험에서 사용하게 될 CNN모델에 관한 설명을 한다. CNN의 모델 구조는 크게 지역 수용 영역, Convolutional Layer, Pooling Layer, 마지막으로 Fully Connected Layer로 구성된다. 본 실험의 지역 수용 영역은 데이터 세트로 변환된 패킷 Payload와 label을 입력받게 된다. 패킷은 이미지 $N \times N$의 형태로써 지역 수용 영역에 (-1, N, N, 1)의 모양으로 입력이 된다. 이 후 Convolutional Layer를 거치게 되는데, Convolutional Layer의 각각의 뉴런이 편향이나 영역과 연결된 5×5의 가중치를 갖고 있다. 이 가중치와 편향은 모든 뉴런에서 사용한다. 이가 의미하는 바는, 입력 데이터마다 위치가 변경 되도 첫 번째 은닉 레이어의 모든 뉴런은 같은 특징을 인식한다는 것을 뜻한다. 실험에서는 두 개의 Convolutional Layer를 거치는데, 첫 번째, 그리고 두 번째 입력 은닉 레이어의 가중치와 편향의 모양은 (5, 5, 1, 32), (5, 5, 32, 64)와 같다. Convolutional Layer를 지날 때마다 샘플링(Pooling) 과정도 거치게 되는데, 실험에서는 2×2 영역을 하나의 뉴런으로 만들게 된다. 최종적으로 샘플링 된, $N \times N$의 패킷 이미지 처리에 사용할 Fully Connected Layer의 모양은 ((N/2)/2, (N/2)/2, 64, 1024)이다. '1024'의 의미는, 두 번째 은닉 레이어를 거치면서 변환된 값이 1024개의 뉴런과 Fully Connected Layer를 형성한다는 뜻이다 [5].

5. 실험 평가

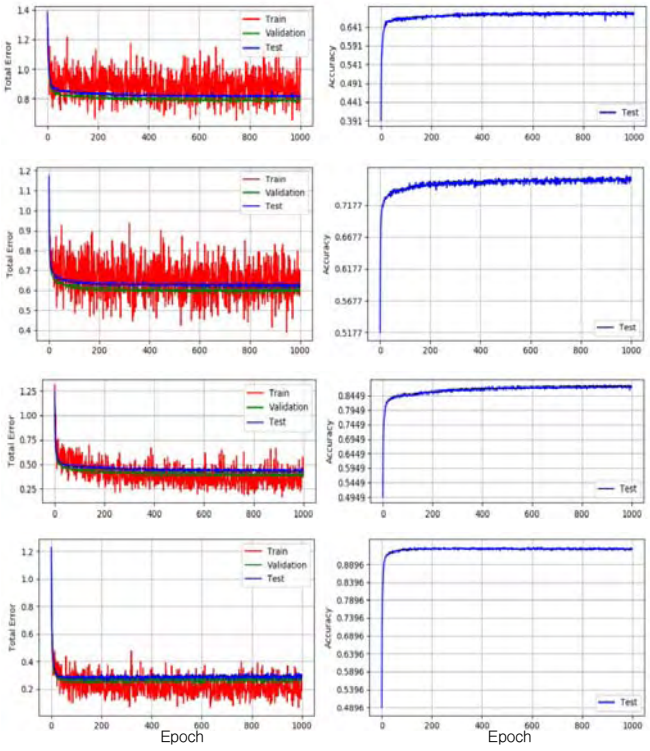
2장과 3장의 트래픽 데이터 정제 및 이미지 변환 작업을 거친 데이터 세트를 CNN 모델에 Feeding을 하여 훈련시켰다. 패킷 단위의 이미지 데이터는 $N \times N$의 정방행렬로써, 그 크기 여하에 따라 얼마만큼의 bit데이터를 사용할지 결정할 수 있다. 이러한 특성에 의해 본 장의 실험에서는, 두 가지에 초점을 맞추었다. 첫 번째는 패킷 이미지의 축의 길이를 조절하는 것이었고, 두 번째는 패킷 이미지의 하나의 문자 데이터, 즉 한 픽셀을 구성하는 원소의 크기를 조절하는 것이었다.

5-1. 패킷 이미지 크기에 따른 실험 및 평가

패킷의 이미지 크기를 조절한다는 것은, 하나의 패킷이 담고 있는 Payload의 크기를 조절한다는 말과 같다. 예를 들어, 8×8의 패킷 이미지는 $64 \times 4bit$의 크기를 갖는다. 즉 패킷 Payload의 256bit(32 Bytes)까지의 데이터를 하나의 패킷 이미지 데이터로 사용한다는 것이다. 본 실험에서는, 패킷 이미지 데이터를, 8×8, 16×16, 32×32, 64×64로 각기 다른 크기로 데이터 세트를 변환하여 CNN 모델에 학습을 시키고 평가를 진행하였다. 그 결과 <표 3>과 같은 결과가 나왔고, 패킷 이미지의 크기를 늘릴수록 높은 정확도를 보였다(그림 4).

<표 3> 5-1 Test Accuracy(%)

	8×8	16×16	32×32	64×64
데이터 크기 (bytes)	32	128	512	2048
전체 평균	67.99	76.20	87.76	93.58
RDP	86.59	88.57	89.59	94.70
Skype	88.19	72.91	95.00	96.67
SSH	48.54	85.56	86.58	88.71
Bittorrent	83.97	77.40	86.05	91.25
Web	33.18	55.21	79.96	95.27



(그림 4) 패킷 이미지 크기에 따른 실험 결과 그래프 [8×8, 16×16, 32×32, 64×64]

훈련 데이터의 Error Value는 각 Epoch마다 100개의 Batch 데이터에 대한 Error Value이고, Validation Error와 함께 전반적으로 하향 곡선을 그리는 것을 결과 그래프(그림 4)를 통해 확인 할 수 있다. 독립적인 패킷을 이미지로 만든 데이터 세트를 가지고 CNN 모델에 학습을 시킨 결과가 나쁘지 않았고, 패킷 이미지의 크기를 늘렸을 때는 더 좋은 정확도를 보였는데, 이유를 들어보자면, Application Layer의 Paylaod는 해당 응용이 사용하는 특정 프로토콜로써 정보를 주고받게 되는데, 프로토콜의 특성상 형식, 의미론적으로 독립적이며 통신의 동기 과정이 명확하고 구별이 가능하다. 또한 원본 PCAP파일을 분석한 결과, RDP는 RFC 1006 TPKT프로토콜과 ISO 8073 COTP를 사용했는데, 이 프로토콜의 헤더 데이터를 본 실험의 패킷 이미지 데이터 크기로 환산한다면, 288bit에 해당하고, 16×16인 패킷 이미지 하나보다 조금 더 큰 것이다. 이를 비추어 생각해 본다면, 패킷 이미지 데이터의 크기를 늘릴 때, Payload의 프로토콜 헤더 데이터를 더 많이 CNN 학습에 적용시킬 수 있다는 말로 해석할 수 있다.

6. 결론

네트워크 트래픽 데이터를 패킷 단위로 하여 CNN 모델에 학습한 결과 그 정확도가 93.58%에 달했다. 기존의 방식보다는 높다고 할 수 없지만, 결코 낮다고 단언할 수 없는 결과이다. 이유인 즉, 네트워크 트래픽의 flow가 갖는 연속성을 가지지 않는, 단일 패킷의 Payload들만을 CNN 모델 입력 데이터 세트로 삼았기 때문이다. 실험 결과를 통해 패킷 Payload 내부에는 이 전에 파악하지 못했던 특징이 존재 할 수 있다는 가정을 세워볼 수 있다.

향후 연구에서는 패킷 단위가 아닌, 연속성을 가지는 flow를 입력 데이터세트로 하여 RNN 모델에 학습을 시키고자 하는 실험을 진행 할 것이다.

참고문헌

[1] 박진완. (2009) “통계 시그니쳐 기반의 응용 트래픽 분류.” 한국통신학회논문지 제34권 제11호(네트워크 및 서비스), 2009.11, 1234-1244 (11 pages)

[2] F. Risso. “Lightweight, Payload-Based Traffic Classification: An Experimental Evaluation.” Communications, 2008. ICC'08. IEEE International Conference on. IEEE, 2008.

[3] Yann LeCun. (2015) “Deep Learning”, nature, International weekly journal of science.

[4] Universitat Politècnica de Catalunya@Barcelon, Spain, HTTP://www.cba.upc.edu/monitoring/traffic-classification#is-our-ground-truth-for-traffic-classification-reliable-dataset

[5] [국내] Giancarlo Zaccone(2016) 예제로 배우는 텐서플로. acorn, PACKT publishing