Multi-Task Hierarchical Learning Based Network Traffic Analytics

Onur Barut  
Dept. of Electrical and Computer Eng.  
University of Massachusetts Lowell  
Lowell, USA  
Onur_Barut@uml.edu

Yan Luo  
Dept. of Electrical and Computer Eng.  
University of Massachusetts Lowell  
Lowell, USA  
Yan_Luo@uml.edu

Tong Zhang  
Network Platforms Group  
Intel Corporation  
Santa Clara, USA  
tong2.zhang@intel.com

Weigang Li  
Network Platforms Group  
Intel Corporation  
Shanghai, China  
weigang.li@intel.com

Peilong Li  
Dept. of Computer Science  
Elizabethtown College  
Elizabethtown, USA  
lip@etown.edu

Abstract—Classifying network traffic is the basis for important network applications. Prior research in this area has faced challenges on the availability of representative datasets, and many of the results cannot be readily reproduced. Such a problem is exacerbated by emerging data-driven machine learning based approaches. To address this issue, we present (Net)$^2$ database with three open datasets containing nearly 1.3M labeled flows in total, with a comprehensive list of flow features, for the research community. We focus on broad aspects in network traffic analysis, including both malware detection and application classification. As we continue to grow them, we expect the datasets to serve as a common ground for AI driven, reproducible research on network flow analytics. We release the datasets publicly and also introduce a Multi-Task Hierarchical Learning (MTHL) model to perform all tasks in a single model. Our results show that MTHL is capable of accurately performing multiple tasks with hierarchical labeling with a dramatic reduction in training time.

Index Terms—Network Traffic Analytics, Malware Detection, Multi-Task Learning, Hierarchical Labeling, Network Flow Features

I. INTRODUCTION

Recent advances in technology have provided researchers a lot of data available for analysis. In particular, many studies have been published in various fields including Computer Vision (CV), Natural Language Processing (NLP), and lately Network Traffic Analysis (NTA) investigating big data with Artificial Intelligence approaches [1]–[7]. Even though many benchmarks and challenges are available for CV [8] and NLP fields [9], there is a lack of common ground for network traffic analysis for researchers to compare their results with others.

Network traffic analysis has become vital for many applications such as quality of service (QoS) control, resource allocation and malware detection. Payload-based packet inspection (DPI) for NTA has serious limitation due to performance, privacy concerns and ineffectiveness on encrypted traffic. Similarly, port-based approach has become obsolete long ago since newer applications mostly use dynamic port allocation.

On the other hand, an emerging trend is to employ machine learning (ML) models trained on labeled datasets as they do not rely on port numbers or payload [7], [10]–[13]. However, each of these methods specialize on a single use-case and cannot be generalized. Therefore, we introduce a multi-task learning scheme to overcome this issue.

There have been a plethora of research attempting to classify and analyze network flows using a variety of datasets. However, unlike the open datasets such as ImageNet [8] available in computer vision research, it is very difficult to find comprehensive datasets for researchers in networking domain to evaluate their proposed techniques, and for the community to reproduce prior art. Many agree that a comprehensive, up-to-date and open dataset for flow analytics is dispensable for the network research community.

In this paper, we address a common ground for NTA and introduce (Net)$^2$, an open database with a collection of multi-labeled network flows and their features for researchers as a benchmarking platform to evaluate their approaches and contribute to NTA research. In order to be comprehensive, we compile datasets for both malware detection and application type categorization, two representative applications. For malware detection tasks, we introduce a new dataset with raw traffic captures obtained from Stratosphere IPS [14] website with almost 500k network flow samples belonging to 20 different types of malware and benign classes. We also create another dataset for malware detection which is obtained from the raw traffic captures from well-known CICIDS2017 dataset [12].

We generate around 550k flow samples for 7 different malware types and a benign class. Lastly, we introduce non-VPN2016 dataset with around 163k flow samples whose raw traffic data are acquired from ISCX-VPN-nonVPN2016 dataset [13] containing different levels of annotations with a number of classes ranging from 7 to 31.

Accepted by International Conference on Communications (ICC21)  
14-23 June, 2021

1https://github.com/ACANETS/NetML-Competition2020
On top of those data, we implement Multi-Task Hierarchical Learning (MTHL) model to perform all the tasks simultaneously as a single classifier. We host the dataset and release the implementations of the algorithms on GitHub and encourage the community to collaborate and grow the collection, pushing the state of art in ML-driven network traffic classification.

Our contribution in this paper is threefold: Firstly, we present a novel malware detection (Net)$^2$ dataset and curate two other datasets originated from open source packet trace captures (named as CICIDS2017 for malware detection and VPN-nonVPN 2016 dataset for traffic classification) with granulated labels and generate more than 100 features including flow metadata and protocol-specific features. Secondly, we provide an open platform for researchers to address network traffic analytics and evaluate baseline results obtained by RF, SVM and MLP models, facilitating the benchmarking of traffic analytics like ImageNet to CV. Thirdly, we introduce a novel MTHL model for network traffic analysis to perform different tasks with hierarchical labeling.

II. RELATED WORK

A. Malware Detection

KDD-Cup 99 [15] and NSL-KDD datasets [16] are common datasets widely used for malware detection [17], [18]. However, those two datasets are out of date and models trained with those are prone to recently emerged attacks. CICIDS2017 dataset [12] is relatively up-to-date and more suitable for detecting recent types of intrusion. Sharafaldin et al. introduce CICIDS2017 dataset with their results obtained using CICFlowMeter [19] features by comparing several techniques including k-nearest neighbor (kNN), random forest and ID3 and report best F1 score 0.98 for ID3 algorithm and 0.97 for random forest whose executing time is one-third of ID3. Gao et al. [7] compare several machine learning and deep learning techniques including Random Forest, SVM and Deep Neural Networks on NSL-KDD and CICIDS2017 datasets. They perform both binary malware detection and multi-class classification and find out that RF model achieves impressive malware detection accuracy and comparable classification accuracy to other deep learning models in multi-class classification tasks.

Random Forest and Support Vector Machine (SVM) classifiers are two effective ways among many machine learning approaches for network traffic analysis [10]. However, most of the researchers use their own data with their own features for NTA, making it difficult to compare one study to another.

B. Traffic Classification

Conti et al. [20] use Random forest with their own dataset and their own features and achieve 95% accuracy on user action classification. Taylor et al. [4] conclude that using statistical features of flow vectors of raw packet lengths such as minimum, maximum, mean, median, standard deviation on binary SVM and Random Forest results in 95% accuracy, but when using multi-class classifier, performance drops to 42% for SVM and 87% for RF. Lashkari et al. [13] introduce ISCX VPN-nonVPN2016 dataset and analyze it with kNN and C4.5 algorithms.

Many researchers adopt VPN-nonVPN2016 dataset and report their results. For example, Yamansavascilar et al. [21] utilize this dataset and compare the results with their own dataset using the same classes. They report that kNN works best for VPN-nonVPN dataset and random forest produces most accurate results on their own dataset. They also report 2% boost in the accuracy of their own dataset with proper feature selection. Wang et al. [5] use packet-level analysis and use deep learning approach for protocol classification. Later, Yao et al. [6] implement attention-based LSTM (Long Short-Term Memory) to classify protocol types in VPN-nonVPN dataset.

Previous studies show that the demand for analyzing network flow is increasing and there is not a unique solution for application classification or malware detection in network analysis. Therefore, there is an urgent need of a generic set of features that are known to be useful for malware detection and application classification with a more comprehensive and up-to-date dataset.

III. (Net)$^2$ DATA COLLECTION

In this paper, we focus broadly on flow classification tasks for various purposes and at different granularity. We study both malware detection and more general flow type identification. We use “top-level”, “mid-level” and “fine-grained” to denote the granularity levels. At "top-level", malware detection is a binary classification problem, i.e. benign or malware, while for flow identification task, the major application categories such as chat, email, stream etc. are annotated. The "mid-level” granularity identifies specific applications such as facebook, skype, hangouts etc. for normal traffic and specific types of malware including Adload, portScan etc. Finally, the “fine-grained” level performs multi-class classification to identify the detailed types of application traffic such as facebook_audio, skype_audio etc. For malware detection datasets, we do not have fine-grained annotations.

A. Dataset Preparation

We start our work by obtaining publicly available raw packet traces from Stratosphere IPS [14] and Canadian Institute of Cybersecurity (CIC) [12], [13]. CIC provides several different types of dataset for Intrusion Detection Systems (IDS) along with an application type classification dataset. Stratosphere IPS only focuses on numerous benign and malware captures. As most of the traffic analysis research resort to extract features from flows or raw packets, we aim to process these packet traces to present a comprehensive list of features including Metadata such as the number of packets and bytes, source and destination port numbers, time duration, time-interval between packets of the flow and protocol-specific features including TLS, DNS and HTTP. Table I give the trace files used to obtain the datasets and the number of flows for each label. We utilize our own accelerated tool to compute these flow features and anonymize the data when making them available.
TABLE I: Number of flow per class for each training set

| line | mid            | top          | # flows |
|------|----------------|--------------|---------|
|      | Adload         | Malware      | 15289   |
|      | Artemis        | Malware      | 8238    |
|      | Benign         | Benign       | 76029   |
|      | BitCoinMiner   | Malware      | 4083    |
|      | CCleaner       | Malware      | 125637  |
|      | Cobalt         | Malware      | 1379    |
|      | Downware       | Malware      | 3857    |
|      | Dridex         | Malware      | 9208    |
|      | Emotet         | Malware      | 45907   |
|      | HTBot          | Malware      | 30442   |
|      | MagicHound     | Malware      | 31485   |
|      | MinerTrojan    | Malware      | 4741    |
|      | PUA            | Malware      | 6171    |
|      | Ramnit         | Malware      | 400     |
|      | Tinba          | Malware      | 18627   |
|      | TrickBot       | Malware      | 37276   |
|      | Trickster      | Malware      | 57800   |
|      | TrojanDownloader| Malware     | 4029    |
|      | Ursnif         | Malware      | 8482    |
|      | WebCompanion   | Malware      | 15776   |

We process the raw packet traces by following the steps depicted in Figure 1 to generate dataset splits for $(Net)^2$ database. We design a feature extraction library to compute the flow features by taking up to first 48 packets in each direction from the raw traffic capture file. Extracted features are stored in JSON format, and features for each flow sample is listed line by line in the output file. Metadata features such as number of packets, source port, destination port etc. can be extracted for every flow sample. However, protocol-specific features such as TLS, DNS and HTTP can be extracted only if the flow sample contains packets for the given protocols. Therefore, some of the flows contain protocol-specific features while all of the flows contain Metadata features. After feature extraction, we mask the source and destination IP addresses. Later, we add time_length as a new feature which is obtained by subtracting time_start from time_end. Then, we remove time_start and time_end from the feature set. These steps are necessary because of two reasons: (1) exposed IP address will reveal anonymity and (2) IP address as well as time features will cause a bias in the prediction due to the data collection setup. After that, "id" and "label" for each flow are added to obtain the $(Net)^2$, CICIDS2017 and VPN-nonVPN 2016 datasets. Finally, generated datasets are split into training, test-std and test-challenge subsets using 8:1:1 ratio, respectively. Labels for training set are open to public; however, labels for test-std and test-challenge sets are reserved to evaluate the models of the participants.

B. Malware Detection Datasets

1) $(Net)^2$ dataset: This dataset is created for malware detection task by obtaining raw traffic data from Stratosphere IPS. To obtain a dataset which has a similar number of flows with previously released CICIDS2017 [12], we utilize 19 different malware 7 different benign trace files. Different files contain different malware types; therefore, each file is annotated with its malware type for mid-level annotations. In addition to that, each malware type is also annotated as Malware for top-level annotations.

2) CICIDS2017: This is a set of raw traffic captures for different types of malware attacks and normal flows [12]. Each trace file contains the whole network traffic throughout the day and 5 traces for each day of a week is collected by CIC. However, only some specific time intervals with packet exchange between the predetermined IP addresses contain the flows we want to extract. Therefore, we download CICIDS2017 dataset from the original source and filter the flows-of-interest according to the descriptions in the dataset website, and extract flow features using the flow feature extraction tool. Similar to the $(Net)^2$ dataset, top-level annotations in CICIDS2017 contain benign or malware classes while mid-level annotations contain 7 different malware types and a benign class.
C. Traffic Classification Dataset

The third dataset focuses on application classification and is called VPN-nonVPN 2016. Similar to CICIDS2017, raw traffic capture files are obtained from the CIC website. As each file is named with action performed with the application such as skype_audio, facebook_video etc. we can label those flows in fine-grained annotations to identify 24 different lower-level classes in many applications. Mid-level annotations contain 18 type of applications (facebook, skype etc.) while top-level annotations are 7 general grouping of those traffic capture data.

VPN-nonVPN dataset contains fine-grained labels whereas (Net)^2 and CICIDS2017 datasets do not reveal lower level actions. Extracted flow numbers from the three datasets are given in Table I.

IV. METHODOLOGY

In this section we briefly explain the flow features for the training sets of all three datasets. From now on, we only use Training set to conduct our analysis and experiments as only this split will be publicly available with the annotations. We extract four set of features: (1) metadata, (2) TLS, (3) DNS, (4) HTTP. Metadata features are protocol-independent features such as number of packets, bytes inbound and bytes outbound, time length of a flow etc. On the other side, there are protocol-specific features such as TLS, DNS and HTTP. Number of ciphersuites and extensions supported by the client or server are a subset of the TLS features. Similarly, DNS query name and DNS answer IP can be given as example for DNS features. Finally, HTTP code and HTTP method are two examples for HTTP features. While metadata features can be extracted for any kind of flow, protocol-specific features can only be extracted if the given flow contains packets with one of these protocols. Therefore, Table II summarizes number of flow features for each training set.

| Dataset            | Metadata | TLS    | DNS     | HTTP    |
|--------------------|----------|--------|---------|---------|
| (Net)^2            | 580858   | 135739 | 114522  | 52185   |
| CICIDS 2017        | 441897   | 75163  | 93568   | 40314   |
| VPN-nonVPN 2016    | 146041   | 19362  | 793     | 52185   |

The imbalance of number of samples in a dataset is an important but mostly inevitable. As an example, in Table I we observe that facebook_audio, hangouts_audio, skype_audio and skype_file classes are oversampled in the VPN-nonVPN 2016 dataset. Similarly, CCleaner in (Net)^2 and portScan in CICIDS2017 datasets reveal the same behavior. Such significantly imbalanced class distribution in the training set may introduce a bias to the predictions of the trained models. However, we leave the datasets as they are to offer a challenge for the researchers to tackle with the imbalanced data.

A. Metadata Features

Metadata features are mostly composed of statistical features or histograms computed from a flow. Metadata features can be extracted for any flow samples. In this study, 31 different metadata features are extracted using the flow feature extraction tool. For the histogram-like features such as compact histogram of payload lengths pld_cnt[], and compact histogram of header lengths hdr_cnt[], the returned values are a constant size of array. Unlike histogram arrays, the other metadata features such as number of bytes inbound bytes_in or source port number src_port return a single value.

B. Protocol-Specific Features

Apart from metadata features, TLS, DNS and HTTP protocols are three other sources of features extracted by the feature extraction tool. Several client and server-based TLS-specific features are extracted in our study. Some of these TLS features are the list of offered ciphersuites, the list of advertised extensions and the key exchange length advertised by or and server. In total, 14 TLS features are extracted.

Our feature extraction tool returns several DNS query and answer features. Similar to TLS features, there are both single-valued and array-like features in DNS involved flows. For example, dns_query_cnt and dns_answer_cnt features are returned as a single integer while the other query and answer features are returned as array of integers for dns_answer_ttl and strings for the rest.

For each flow containing HTTP packets, we extract six different HTTP features. http_method, http_code and http_content_len are the only features which are represented as a single integer value while the others contain string datatype which needs to be processed before classification.

C. Design of Multi-Task Hierarchical Learning

The structure of MTHL model is provided in Figure 2. We now discuss the blocks in detail.

1) Preprocessing: After we obtain the training set split as depicted in Figure we need to convert the JSON format data into a two-dimensional array whose columns represent features and rows stand for flow samples. Firstly, we discard source and destination IP address features as they are already masked. All other metadata features in the datasets contain numeric values and hence it is straightforward to import these datasets into a matrix form that will be used to train the classifiers. For array-like features such as hdr_cnt[], we
just treat each of those dimensions in this array as a separate feature such as $hdr_{cnt_0}$, $hdr_{cnt_1}$ etc. up to $hdr_{cnt_k}$ for $k+1$ dimensional $hdr_{cnt[k]}$ feature and place each of those into separate columns in the final data matrix. Following this approach yields 121 columns obtained from 31 flow metadata features. Finally, the data are normalized so that data in each feature column follows standard normal distribution.

2) Input Block: Batch normalization help the deep model train in a more stable and faster way by normalizing the input batch for the next layer. Therefore, we use batch normalization before any operation in the network. One dimensional convolutional layer is implemented to exploit the spatial correlation between the elements of histogram-like features after batch normalization because these elements are highly correlated. After convolution, Rectified Linear Unit (ReLU) activation function is adopted as it simplifies training the model. Finally, another batch normalization is applied to keep the training stable. Stride of convolution kernel is set to 2 with zero-paddling and 16 activation maps are obtained in the input block, yielding 61x16 tensor for a single input flow data.

3) Residual Block: Residual connections enables training deeper models and improve the performance. Therefore, instead of a plain neural network, the residual block is implemented whose details are depicted in Figure 2. Residual block includes one dimensional convolution, ReLU activation followed by batch normalization, and a residual connection. In the first part, the data shape is preserved as (61x16) in the convolution operations with stride as 1. In the second part, spatial size of the data is halved with stride as 2 and the number of activation maps are doubled from 16 to 32. For that purpose, 1D convolution layer is also applied in the residual connection. The last part of the residual block is very similar to the first part. It takes (32x32) tensor as input and outputs the same shape.

4) Output Block: This block is where the model has multiple branches for different levels of classifications. Both of the branches apply 1D convolution with stride 2 to further reduce the spatial size and increase the number of activation maps, yielding 16x64 tensor for each flow. After ReLU activation and batch normalization, two-dimensional data is reshaped to one dimension for final classification. Application and malware type classification task, which uses the mid-level annotations, is performed using fully connected layer with softmax activation. On the other hand, top-level classification branch, additional fully connected layer is adopted to enable the connection from the mid-level classification, thus ensuring the hierarchy.

V. EXPERIMENTAL EVALUATION

In this section, we demonstrate how the proposed granularity of labels in these datasets can be used for Multi-Task Hierarchical Learning (MTHL) to combine all different tasks in a single model and compare its performance with the baselines for flow analysis.

A. Experiment Setup

Application and malware type classification tasks are defined as mid-level classification while protocol identification and malware detection are performed as top-level classification tasks. All the flow data from VPN-nonVPN dataset is used with both top-level and mid-level annotations. Since these flows belong to normal traffic, only the malware flows are adopted from $(Net^2)$ and CICIDS2017 datasets with both mid-level and top-level annotations for the MTHL model.

To evaluate the proposed MTHL model, we prepare four baseline machine learning models for comparison. We utilize Scikit-learn library for traditional machine learning classifier training on Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz and Keras library with Tensorflow backend for MTHL model training on NVIDIA GeForce GTX 1060 GPU with 6 GB memory and CUDA version 10.2.

80% of the training split is used to train the classifiers while the remaining 20% is reserved for evaluation and referred as validation set. Macro-average F1 score is utilized to evaluate the performance since it provides more general insight for multi-class classification problems with imbalanced data. Adam optimization with learning rate 0.001 and cross-entropy are adopted for loss evaluation and training. Batch size is set to 200 and 100 the model is trained for 100 epochs.

B. Baselines

Random Forest, k-Nearest Neighbor (kNN), Support Vector Machine (SVM) and Multi-layer Perceptron (MLP) are implemented as baselines since they are easy to train and widely used in many applications. As explained in Section IV, protocol-specific features are not available for all of the flows in the datasets. Therefore, we choose only metadata features to expedite the efficiency of the proposed classifiers.

C. Results

We provide the classification results of 6 different scenarios obtained using validation set and summarized in Table III. Each row gives a different task with the given dataset and the level of the annotation. We observe that different baseline models perform best for each scenario. For example, RF model is the best for application classification dataset while kNN classifier performs the best for $(Net^2)$ -t and $(Net^2)$ -m scenarios. Even though CICIDS2017 is another dataset for malware classification, unlike $(Net^2)$, the best performing baseline model is MLP for both of the cases. This indicates that there is not a single baseline model that performs best for all the tasks. Moreover, different models would be the best for different tasks, as in our case. However, our proposed MTHL model is capable of executing all the tasks simultaneously and it outperforms the best performing baseline models except for the two tasks. Our model could produce 0.639 F1 score for malware type classification task in $(Net^2)$ dataset, which is 0.023 higher than SVM model but 0.060 lower than the best performing kNN classifier. Similarly, our model has some limitations while performing application-level classification. RF classifier, which is the best baseline for VPN-nonVPN2016-m
TABLE III: F1 scores for different scenarios. t: top-level, m: mid-level classification task

|                | RF  | kNN | SVM | MLP | MTHL |
|----------------|-----|-----|-----|-----|------|
| (Net)^2 - t    | 0.984 | 0.984 | 0.939 | 0.964 | 0.994 |
| (Net)^2 - m    | 0.655 | 0.699 | 0.616 | 0.639 | 0.639 |
| CICIDS2017-t   | 0.978 | 0.989 | 0.989 | 0.989 | 0.994 |
| CICIDS2017-m   | 0.890 | 0.928 | 0.880 | 0.952 | 0.955 |
| VPN-nonVPN2016-t | 0.646 | 0.581 | 0.404 | 0.456 | 0.661 |
| VPN-nonVPN2016-m | 0.625 | 0.576 | 0.425 | 0.388 | 0.616 |

The results in Table III also show that the proposed MTHL model can produce more accurate classification results for top-level classes, however, it is slightly less accurate in mid-level class classification. It is important to note that MTHL model is a single model which is trained once to achieve multiple tasks within a single model while all other baseline classifiers are individually trained for the specified task. Moreover, for each task, different baseline models perform the best while our proposed MTHL model can either outperform or operate with comparable performance. For example, RF and kNN models are the best choices for (Net)^2 - t while kNN is the only best classifier for (Net)^2 - m. Similarly, MLP is the best model for CICIDS2017 top and mid-level classification tasks while RF classifier is the most accurate baseline for VPN-nonVPN 2016 top and mid-level classification tasks. Since there is not a single baseline model which performs the best for all of six tasks, one needs to train six tasks with four baselines, making 24 different training time that can last up to three days of computation. On the other hand, training our MTHL model would take only a few hours on GPU while ensuring a higher performance for four of six tasks. Therefore, we can conclude that MTHL model is capable of learning common representations for different tasks and able to correctly classify the network traffic with different granularity.

VI. CONCLUSION AND FUTURE WORK

To enable data driven machine learning based research in network flow analytics, we introduce (Net)^2 datasets curated from open-sources for network traffic classification. We release the flow features and different levels of annotations, aiming to present a common dataset for research community. We also introduce a novel multi-task hierarchical learning model and show that our model can be trained for multiple tasks with different label granularity and achieve higher performance for four of six tasks with much shortened training time when compared to baseline machine learning classifiers. Our results have inspired potential future work including, utilizing TLS, DNS and HTTP flow features as well as raw bytes of the traffic data to achieve higher accuracy. We expect that the (Net)^2 datasets will promote machine learning based network traffic analytics and establish a common ground for researchers to propose and benchmark their approaches in this AI era.

ACKNOWLEDGMENTS

This work was supported in part by Intel Corporation.

REFERENCES

[1] P. Wang, S. Li, F. Ye, Z. Wang, and M. Zhang, “Packetgan: Exploratory study of class imbalance for encrypted traffic classification using cgan,” in ICC 2020 - 2020 IEEE International Conference on Communications (ICC), 2020, pp. 1–7.
[2] O. Barut, L. Zhou, and Y. Luo, “Multi-task lstm model for human activity recognition and intensity estimation using wearable sensor data,” IEEE Internet of Things Journal, pp. 1–20, 2020.
[3] O. Barut and A. A. Alatan, “Geospatial object detection using deep networks,” in Earth Observing Systems XXIV, J. J. Butler, X. J. Xiong, and X. Gu, Eds., vol. 11127, International Society for Optics and Photonics. SPIE, 2019, pp. 15–23. [Online]. Available: https://doi.org/10.1117/12.2530027
[4] V. F. Taylor, R. Spolao, M. Conti, and I. Martinovic, “Appsscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic,” in 2016 IEEE European Symposium on Security and Privacy (EuroS P’16), March 2016, pp. 439–454.
[5] P. Wang, F. Ye, X. Chen, and Y. Qian, “Dataten: Deep learning based encrypted network traffic classification in sdn home gateway,” IEEE Access, vol. 6, pp. 55 380–55 391, 2018.
[6] H. Yao, C. Liu, P. Zhang, S. Wu, C. Jiang, and S. Yu, “Identification of encrypted traffic through attention mechanism based long short term memory,” IEEE Transactions on Big Data, pp. 1–1, 2019.
[7] M. Gao, L. Ma, H. Liu, Z. Zhang, Z. Ning, and J. Xu, “Malicious network traffic detection based on deep neural networks and association analysis,” Sensors, vol. 20, p. 1452, 03 2020.
[8] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255.
[9] SENSEVAL ’01: The Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems. USA: Association for Computational Linguistics, 2001.
[10] I. Ahmad, M. Basheri, M. J. Iqbal, and A. Rahim, “Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection,” IEEE Access, vol. 6, pp. 33 789–33 795, 2018.
[11] B. Anderson, S. Paul, and D. McGrew, “Deciphering malware’s use of tls (without decryption),” Journal of Computer Virology and Hacking Techniques, 07 2016.
[12] I. Sharafaldin, A. Habibi Lashkari, and A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” 01 2018, pp. 108–116.
[13] A. Habibi Lashkari, G. Draper Gil, M. Mamun, and A. Ghorbani, “Characterization of encrypted and vpn traffic using time-related features,” 02 2016.
[14] Stratosphere, “Stratosphere laboratory datasets,” 2015. [Online]. Available: https://www.stratosphereips.org/datasets-overview
[15] KDD Cup 1999 Data, 1999. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[16] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, “A detailed analysis of the kdd cup 99 data set,” IEEE Symposium. Computational Intelligence for Security and Defense Applications, CISDA, vol. 2, 07 2009.
[17] J. Zhang, F. Li, and F. Ye, “An ensemble-based network intrusion detection scheme with bayesian deep learning,” in ICC 2020 - 2020 IEEE International Conference on Communications (ICC), 2020, pp. 1–6.
[18] L. Ma, Y. Chai, L. Cui, D. Ma, Y. Fu, and A. Xiao, “A deep learning-based ddos detection framework for internet of things,” in ICC 2020 - 2020 IEEE International Conference on Communications (ICC), 2020, pp. 1–6.
[19] A. Habibi Lashkari, G. Draper Gil, M. Mamun, and A. Ghorbani, “Characterization of tor traffic using time based features,” 01 2017, pp. 253–262.
[20] M. Conti, L. V. Mancini, R. Spolao, and N. V. Verde, “Analyzing android encrypted network traffic to identify user actions,” IEEE Transactions on Information Forensics and Security, vol. 11, no. 1, pp. 114–125, Jan 2016.
[21] B. Yamansavasical, M. A. Guvensan, A. G. Yavuz, and M. E. Karsiligil, “Application identification via network traffic classification,” in 2017 International Conference on Computing, Networking and Communications (ICNC), Jan 2017, pp. 843–848.