畳み込みネットワークを使用するサイバー攻撃検知システムにおける特徴の順序の検出性能への影響について

何 鵬挙 [1,a)]　馮 尭かい [2,b)] 櫻井 幸一 [3,c)]

概要：近年，研究者たちは IoT セキュリティを巡り，AI 技術に基づき，いろいろな IoT-IDS(Internet of Things – Intrusion Detection System)を提出した．畳み込みネットワーク（CNN)や回帰型ニューラルネットワーク(RNN)を利用する検知システムが多く提案されている．例えば，CNN-GRU,CNN-LSTM など．しかしながら，これらの提案では，利用する特徴の順序は検知性能に対して何のような影響を与えるかはまだ調べられていない．本研究では，我らは MQTT-IoT-2020 データセットを利用し，同じモデルとパラメータを利用し，用いる特徴の順番を調整しながら，検知性能を観察した．一番高い検知精度はおよそ 82 パーセント，一番低いのはおよそ７３パーセントで，検知精度の差はおよそ 10 パーセントであることが分かった．
キーワード：機械学習　ネットワークセキュリティ　侵入検出・検知

A Research : Influence of Order of Features in Cyber Attack Detection System

PENGJU HE[1,a)]　YAOKAI FENG[2,b)]　KOUICHI SAKURAI[3,c)]

Abstract: In recent years, researchers published many kinds of Internet of Things – Intrusion Detection System(IoT-IDS), which many of these IoT-IDSs are based Convolution Neural Network(CNN) or Recurrent Neural Network(RNN). However, in their papers, the order of features have not been researched if it could influence the performance of the models. In this paper, To find out the order of features in CNN-LSTM(Long Short-Term Memory), we use MQTT-IoT-2020 dataset and change the order of features to observe the difference of accuracy of models.

Keywords: Machine Learning Network Security Intrusion Detection System

1. はじめに

近年，AI 技術の発展に伴い，いろいろな Intrusion Detection System(IDS)は研究者たちに提唱された[1]．ネットワークセキュリティ領域の場合には，AI に基づく IDS を使用した場合，訓練された IDS モデルは悪性と良性パケットを識別できる．もし攻撃があれば，管理者と管理システムに警報し，防御される．しかしながら，これらの提案は利用する特徴の順序は検知性能に対して何のような影響を与えるかはまだ調べられていない．

本論文では,我らは MQTT-IoT-2020 データセット[2]を利用し，同じモデルとパラメータを利用し，用いる特徴の順番を調整しながら，検知性能を観察した．その後は結論の原因を説明し，最後で自身の今後の研究の進展については述べる．

2. 本研究の研究目標と研究方法

本論文では，注目されるのは主に IoT―IDS である．IoT 環境では使用されるプロトコルが伝統的なデバイスに比べたら，アプリケーション層でのプロトコルで今よく使われるのは Message Queuing Telemetry Transport（MQTT）[11]プロトコル．その原因は計算能力．MQTT プロトコルは軽量で，柔軟性高くで，IoT デバイスにとって相応しいプロトコルである．それで，我らは MQTT―IoT―2020 を使用し，モデルをトレーニングする．

精度は特徴の順番に影響されるかを見つけるため，我らは CNN―LSTM ハイブリッドモデルを使用し，異なる特徴の順番でトレーニングし，精度一番高い組み合わせと精度一番低い組み合わせを記録する．それで，実験を何回も繰り返した上で，結果を提出する．

(1) 九州大学　システム情報科学府
Kyushu University Graduate school and Faculty of Information Science and Electrical Engineering
(2) 九州大学　システム情報科学研究院
Kyushu University
(3) 九州大学　システム情報科学研究院
Kyushu University

a) hepj1999@gmail.com
b) fengyk@ait.kyushu-u.ac.jp
c) sakuraicsce2009g@gmail.com

3. モデルとトレーニング

3.1 IDS

Intrusion Detection System（IDS）は不正または異常な通信を検知し，管理者に通知するシステムで，主に検知パターンや検知のルールを用い，不正な通信を検知する．

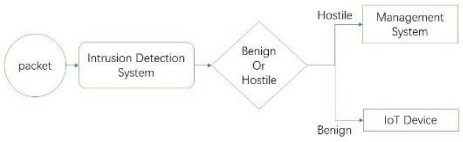

Figure 1 IoT-IDS Simple Structure

侵入検知システムは攻撃を検知するため，ネットワーク経路でのパケットを検査することができる．

IDS 領域で AI を使用される原因の一つはルールやすでに分かった特徴をセキュリティエキスパートに決められる．しかしながら，人類はすべてのパラメータを間違いなく，設定することは無理である．AI を使用すると，また分からない特徴やルールを学習し，見たことのない攻撃を識別できる可能性もある．

3.2 データセット紹介

このデータセットの中でパケットのデータを利用した．今回の目標は Flow ではなく，パケットである．攻撃のタイプは DoS 攻撃，Scan_A 攻撃，Scan_U 攻撃，Sparta 攻撃である[2]. 全部の特徴は 30 種類である，しかしながら，今回の実験では，全部の特徴を使用ではなく，プロトコルの中で意味ある攻撃にとって意味がある可能性がある特徴を使用する．例えば，パケット向けの IDS にとって，Timestamp という特徴は意味がいない，別々に識別するので, それとも, [2]で, 著者のコードの中で Timestamp, Source の IP と Destination の IP を省略した．

図 2 は全部の特徴とラベルである．データストラクチャは大体 int64，float64 とオブジェクトである．オブジェクトは今回で使い難いので，実験で省略した．

今回の実験では，使用された特徴は 22 種類である．ttl，ip_len，ip_flag_df，ip_flag_mf，ip_flag_rb，tcp_flag_res，tcp_flag_ns，tcp_flag_cwr，tcp_flag_ecn，tcp_flag_urg，tcp_flag_ack，tcp_flag_push，tcp_flag_reset，tcp_flag_syn，tcp_flag_fin，mqtt_messagetype，mqtt_messagelength，mqtt_flag_retain，mqtt_flag_qos，mqtt_flag_willflag，mqtt_flag_clean，mqtt_flag_reserved．以上は全部の特徴である．One-hot エンコードするため，Y は四つの次元に分かれ，具体的な形式は以下になる．

[0, 0, 0, 1]

この List の意味は，Sparta 攻撃である．One-hot エンコーディングの使用は分類問題に重要な部分である．

```
Data columns (total 35 columns):
 #   Column               Dtype
---  ------               -----
 0   timestamp            object
 1   src_ip               object
 2   dst_ip               object
 3   protocol             object
 4   ttl                  float64
 5   ip_len               float64
 6   ip_flag_df           float64
 7   ip_flag_mf           float64
 8   ip_flag_rb           float64
 9   src_port             int64
 10  dst_port             int64
 11  tcp_flag_res         float64
 12  tcp_flag_ns          float64
 13  tcp_flag_cwr         float64
 14  tcp_flag_ecn         float64
 15  tcp_flag_urg         float64
 16  tcp_flag_ack         float64
 17  tcp_flag_push        float64
 18  tcp_flag_reset       float64
 19  tcp_flag_syn         float64
 20  tcp_flag_fin         float64
 21  mqtt_messagetype     float64
 22  mqtt_messagelength   float64
 23  mqtt_flag_uname      float64
 24  mqtt_flag_passwd     float64
 25  mqtt_flag_retain     float64
 26  mqtt_flag_qos        float64
 27  mqtt_flag_willflag   float64
 28  mqtt_flag_clean      float64
 29  mqtt_flag_reserved   float64
 30  is_attack            float64
 31  DDoS                 float64
 32  scan_A               float64
 33  scan_sU              float64
 34  sparta               float64
```

Figure 2 Features and Labels

3.3 CNN

畳み込みニューラルネットワーク(CNN)はディープラーニングでよく使われる一つのモデルである．CNN は動物の視覚野から発想を得て福島邦彦によって提唱されたネオコグニトロンに起源を持つ[3]. AI 技術の発展に伴い，CNN は画像認識や音声認識などいろいろな領域に利用された．

CNN と単純な多層パーセプトロンより，畳み込み部分，即ち特徴の抽出部分があり，過学習を避け，トレーニングで計算必要なパラメータも下がる．つまり，畳み込みはデータを次元を圧縮することができる． IoT 環境で，特徴とトレーニングするパラメータが少なくなると，必要な計算能力とコストが低くなる．

以下は CNN の基本操作である．

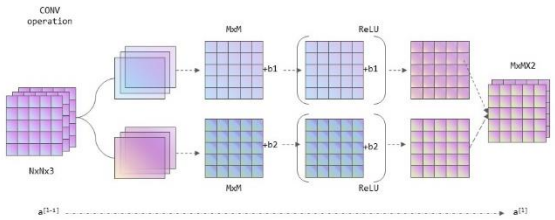

Figure 3 CNN Conv2d

今回は Conv1d 層を利用した．画像認識とちょっと違いがあり，Maxpooling 層は特徴を直接に省略したので，今回は使用しなかった．

3.4 LSTM

回帰型ニューラルネットワーク（RNN）は 1986 年提唱された[4]．今 RNN の一つの長短期記憶（LSTM）[9]は各領域でよく使われる．

バニラな RNN に比べると，LSTM は忘却ゲートがあるので，RNN の勾配消失問題を解決する．LSTM は時間の流れを伴い，記憶を持つ可能性がある．

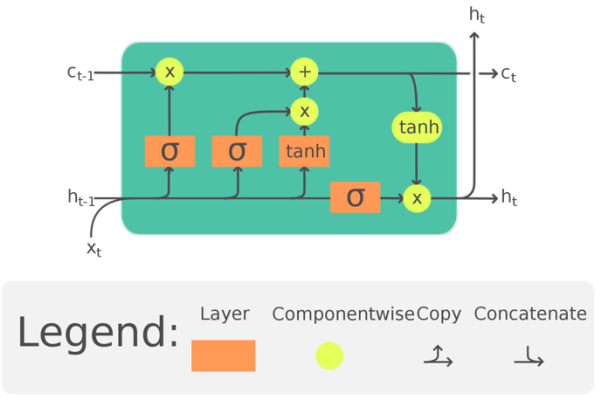

Figure 4 Simple LSTM Structure[12]

IoT-IDS では，パケットの特徴の順番は規律性がある，例えば，Transmission Control プロトコル（TCP）のフラグ値と TCP パケットの長さ，接続用の TCP パケットの長さは短いから，もし長い接続用のパケットを受けると，変なパケットを認める可能性がある．もし RNN を使用すると，その規律性を学習する可能性がある．

3.5 組み合わせとトレーニング

利用された API は Tensorflow[7]と Keras[8]である．

CNN-LSTM は複雑な組み合わせを使用すると，精度は高くすることもある[5][6]. 今回我らが使用したモデルの構造は四つの CNN と一つの LSTM にする．図 5 はモデルの構造である．今回ではシンプルな構造を使用するのは，IoT 環境で相応しいシンプルなモデルを構築するためである．

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 20, 32)	128
conv1d_1 (Conv1D)	(None, 18, 64)	6208
conv1d_2 (Conv1D)	(None, 16, 128)	24704
conv1d_3 (Conv1D)	(None, 14, 64)	24640
lstm (LSTM)	(None, 4)	1104
dense (Dense)	(None, 4)	20

Total params: 56,804
Trainable params: 56,804
Non-trainable params: 0

Figure 5 5 Model Structure

プリプロセッシング階段で，四つの攻撃があるので，One-hot を利用し，エンコードした，Y は四次元のベクトルになった．それで，X を正規化したいが，実験した結果は良くなかった，精度は正規化しない場合より下がった．

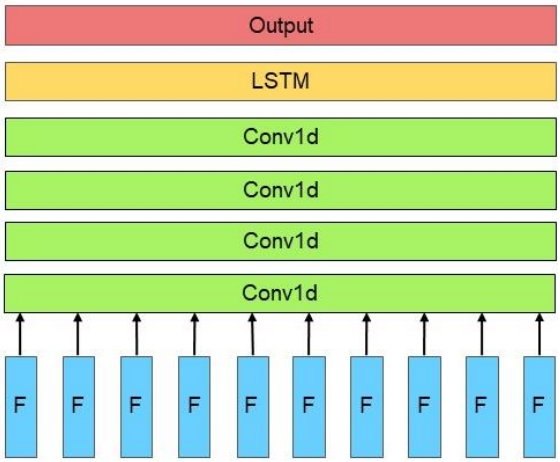

Figure 6 Detailed Model Structure

モデルをコンパイルした時，最適化関数は Stochastic Gradient Descent(SGD)[10]を利用した，今回の目標は特徴の順番の効果を見つけるので，学習率は統一した．各学習率は実験した前に，良い学習率を見つけたので，過学習と過小学習する場合がいないと判断した．しかしながら，SGD とランダムにデータを選択するため，ランダム性があるので，毎回同じパラメータでも得られる結果がちょっと違いがあるので，順番を変わらなく実験をする必要もある．

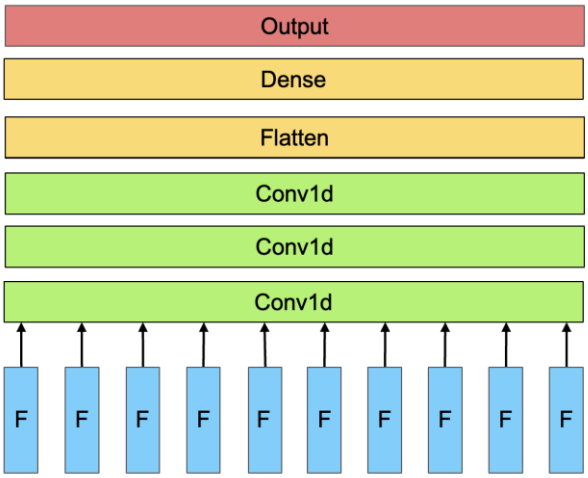

Figure 7 補足するモデル，単純な CNN モデル

毎回データを入力した後，データセットの特徴をシャッフルし，特徴の順番を混じた．二十二種類の特徴があるので，全部の組み合わせは確か22!種類がある，コンピュータでも不可能である．

特徴の順番のシャーフルは二つのプロセスがある．CNN の畳み込みから考えると，まずは元の順番を最初から最後までを動揺する．動揺するアルゴリズムは以下である．

```
i = 0
while i < len(columns) / 2:
    if(i % 2) == 1:
        temp = columns[i]
        columns[i] = columns[len(columns) - 1 - i]
        columns[len(columns) - 1 - i] = temp
    i += 1
```

コード 1　自己シャーフル

```
['ttl',
 'ip_len',
 'ip_flag_df',
 'ip_flag_mf',
 'ip_flag_rb',
 'tcp_flag_res',
 'tcp_flag_ns',
 'tcp_flag_cwr',
 'tcp_flag_ecn',
 'tcp_flag_urg',
 'tcp_flag_ack',
 'tcp_flag_push',
 'tcp_flag_reset',
 'tcp_flag_syn',
 'tcp_flag_fin',
 'mqtt_messagetype',
 'mqtt_messagelength',
 'mqtt_flag_retain',
 'mqtt_flag_qos',
 'mqtt_flag_willflag',
 'mqtt_flag_clean',
 'mqtt_flag_reserved']
```

Figure 8 元の順番

```
['ttl',
 'mqtt_flag_clean',
 'ip_flag_df',
 'mqtt_flag_qos',
 'ip_flag_rb',
 'mqtt_messagelength',
 'tcp_flag_ns',
 'tcp_flag_fin',
 'tcp_flag_ecn',
 'tcp_flag_reset',
 'tcp_flag_ack',
 'tcp_flag_push',
 'tcp_flag_urg',
 'tcp_flag_syn',
 'tcp_flag_cwr',
 'mqtt_messagetype',
 'tcp_flag_res',
 'mqtt_flag_retain',
 'ip_flag_mf',
 'mqtt_flag_willflag',
 'ip_len',
 'mqtt_flag_reserved']
```

Figure 9 コード 1 を実行した後の順番

これで，畳み込みの操作は元の順番と全然違う．

その後，python のランダムモジュール shuffle 関数[13]を使用し，再びシャーフルする．毎回のランダムシードはタイムスタンプで生成するので，毎回のシャーフルは違う順番になる．

3.6 結果

今回は CNN-LSTM のモデルを 50 回以上の実験をした．精度の計算式は以下式 1 になる．

$$acc_{val} = \frac{\#correct\_validation}{\#validation\_setsize} \qquad (式\ 1)$$

各順番を使用したが，検証精度の最大限，最小限，平均数，中央値は以下の表 1 になっていた．

Table 1 各順番の CNN-LSTM バリデーション精度

最大限	最小限	平均数	中央値
0．8206	0．7329	0．7725	0．7660

最大限の差は大体 8．7 パーセントである．

ランダム性を避けるため，同じ順番でも一回をしたが，精度は前回と絶対同じとは言えないが，その差は大体 3 パーセントぐらいであった．

ある順番により，十回ぐらいの実験をした，検証精度の最大限，最小限，平均数，中央値は以下の表 2 になっていた

Table 2 順番を変わらなくのバリデーション精度

最大限	最小限	平均数	中央値
0．8093	0．7871	0．7990	0．7947

実験により，順番を変わらなく，精度も変わったが，その差は変わった順番により小さい，精度の差は特徴の順番に影響されると証明した．

以上は CNN-LSTM モデルの実験と結果である．以下は単純な CNN モデルであるの場合の実験結果である (Table 3).

Table 3 各順番の CNN バリデーション精度

最大限	最小限	平均数	中央値
0．8226	0．7359	0．7763	0．7719

CNN のモデルは十回以上の実験をした，最大限の差も大体 8．7 パーセントである．CNN での特徴の順番に影響されると証明する．

4. おわりに

本論文では，IoT-IDS 向けの CNN-LSTM モデルの特徴の順番は精度に影響されるやされないかを実験で調べた．

最初は AI 技術と IDS の発展を述べ，自分の研究目標と研究方法を紹介した．それで，CNN と LSTM 簡単に紹介し，利用された原因と実験の結果を述べた．補足するため，CNN の実験もした．最後は未来への発送と今後の研究をまとめた．

CNN-LSTM モデルの精度は特徴の順番に影響されたと実験で証明されたが，実際にどんな部分が原因なのか今はまた分かりません．補足実験の部分は CNN の精度は影響されると証明したが，RNN 部分の実験はしないので，今後も実験する必要がある．今推測されたのは CNN や RNN の構造である，畳み込みというプロセスは特徴ベクトルの次元を減らす，もしその組み合わせは良くないと，重要な特徴を直接に減らす可能性がる．しかしながら，もし特徴は確かに精度を影響するなら，一番いい順番を早く見つける方法も今後のテーマになる可能性がある．

今使用したモデルは複雑ではないので，もしもっと複雑なモデルストラクチャを使用すると，精度はもっと高くする可能性がある．しかしながら，IoT-IDS はデバイスの性能のせいで，できるだけ小さいモデルを使用する，そのバランスも重要と考えられる．

参考文献

[1] Kim A, Park M, Lee D H. AI-IDS: Application of deep learning to real-time Web intrusion detection[J]. IEEE Access, 2020, 8: 70245-70261.
[2] Hindy H, Bayne E, Bures M, et al. Machine learning based IoT intrusion detection system: an MQTT case study (MQTT-IoT-IDS2020 dataset)[C]//International Networking Conference. Springer, Cham, 2020: 73-84.
[3] Matsugu M, Mori K, Mitari Y, et al. Subject independent facial expression recognition with robust face detection using a convolutional neural network[J]. Neural Networks, 2003, 16(5-6): 555-559.
[4] Rumelhart D E, Hinton G E, Williams R J. Learning representations by back-propagating errors[J]. nature, 1986, 323(6088): 533-536.
[5] Sun P, Liu P, Li Q, et al. DL-IDS: Extracting features using CNN-LSTM hybrid network for intrusion detection system[J]. Security and Communication Networks, 2020, 2020.
[6] Praanna K, Sruthi S, Kalyani K, et al. A cnn-lstm model for intrusion detection system from high dimensional data[J]. J. Inf. Comput. Sci, 2020, 10: 1362-1370.
[7] https://www.tensorflow.org/
[8] https://keras.io/
[9] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.
[10] https://en.wikipedia.org/wiki/Stochastic_gradient_descent
[11] https://mqtt.org/
[12] https://en.wikipedia.org/wiki/Long_short-term_memory#/media/File:LSTM_Cell.svg
[13] https://docs.python.org/3/library/random.html?highlight=shuffle#random.shuffle