修士論文

ディープラーニングによる IP ネットワーク上の

ストリーミングトラヒック識別の検討

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

1310351　中野　和俊

2017 年 8 月

目次

1. 序論

1.1. 本研究の背景

IP ネットワーク上のアプリケーションのトラヒックを正確に識別することは，帯域の管理やアプリケーションの Quality of Service (以下 QoS)の確保，セキュリティの確保の点で，近年重要性が高まっている．一方，プライバシー保護やデータの安全性確保ため，伝送されるトラヒックが暗号化される割合は増加している．トラヒックが暗号化されることで，現在，主に利用されているトラヒック識別の二つ手法では，識別が困難である，という課題が大きくなってきている．

その二つの手法とは，一つ目は，IP ネットワークを流れるパケットのヘッダ，特にポート番号で識別する方法，二つ目はパケットのペイロードを詳しく解析し特定のシグニチャーを探して識別する方法，である．いずれの手法も，詐称されたポートの使用，暗号化されたチャネルでトラヒックをトンネリング，動的なポートを使用するストリーミング系のアプリケーション，などの課題に対して，正確に識別することができず，信頼性に欠ける．

このような状況で，トラヒックのパケットサイズや到着間隔などの統計情報から抽出した特徴量に基づいて識別する手法が提案されている[4][5][6][7][8][9]．この方法は，ポート番号による識別やペイロード解析による識別などの従来手法の課題に対して，ポート番号など IP パケットのヘッダー情報を使わない，統計情報のため暗号化されている影響を受けにく，ペイロードを参照しないためプライバシーに対応している，という利点がある．しかし精度と，特定の状況やネットワーク環境に依存しない汎化性能の向上ため，より良いアルゴリズムが期待されている．

1.2. 本研究の目的

前節で述べたように，ストリーミングトラヒックなど暗号化された IP ネットワーク上のトラヒックに対しても，高い精度と汎化性能の識別を行うニーズは高まっている．そこで，本研究では，機械学習の 1 手法であるディープラーニングを用いて，IP ネットワーク上のトラヒック識別，特にストリーミングトラヒック識別を，高精度かつ高い汎化性能で実現する手法の検討，提案することを目的とする．

1.3. 本論文の構成

本論文は以下の全 5 章で構成される．

第 1 章は本研究内容を説明する導入のため，研究の背景，目的と論文の構成について述べる．

第 2 章では，トラヒック識別の関連研究について述べる．

第 3 章では，第 2 章で述べた関連研究を踏まえ，ディープラーニングによる IP ネットワーク上のトラヒック識別，特にストリーミングトラヒック識別の検討を行う．ディープラーニング手法を適用する際の課題を明確化し，解決方法を詳しく述べる．

第 4 章で，実験結果とその評価を行う．関連論文との比較し，提案手法の効果を考察する．

第 5 章は結論で，まとめと今後の研究について議論する．

2. 関連研究と課題

本章では，トラヒック識別について，もっとも広く使われている二つの手法と，その問題点について述べる．また，その問題点を解決する手法を提案している関連研究について，概要を述べる．

2.1. 現状のトラヒック識別手法

現状では，ポート番号による識別と，ペイロード解析による識別がもっとも広く利用される手法であるが，メリットとデメリットは表 2.1 の関係にある[1].

表 2.1：従来手法のメリットとデメリット

手法	精度	計算量	暗号化・ストリーミング	プライバシー
ポート番号	30%以下	少	非対応	一部侵害
ペイロード解析	ほぼ 100%	多	非対応	侵害

以上から，暗号化されているトラヒック，ストリーミングトラヒックを含めて，少ない計算量で高い精度を実現し，かつ，プライバシーを侵害しないためにペイロードを参照しない識別方法が必要と言える．

本節では，ポート番号による識別と，ペイロード解析による識別の詳細を述べる．

2.1.1. ポート番号による識別

ポート番号とは，OSI 参照モデルの 7 階層のうちトランスポート層の通信で使われる TCP, UDP による通信を識別するための数字の識別子である．このポート番号のうち，事実上広く使われている番号は，Internet Assigned Numbers Authority (IANA) [2] により，ポート番号と対応するアプリケーションの組み合わせが管理されており，Well-known ポートと呼ばれている．[21]によると，70%以上のトラヒックは Well-known ポートを用いていない動的な通信であるため，ポート番号による識別は，多くのトラヒックで有効に機能しない．

ポート番号による識別の一例として，ネットワーク間の通信を制御する装置として，Firewall が広く使われているが，多くの Firewall ではポート番号による識別を使

って，通信の制御をおこなっている．

このように，Well-known ポートを使った通信では有効に機能するポート番号による識別であるが，動的なポートを使った通信や，暗号化された通信では，ポート番号による識別は行えない．暗号化された通信が識別できない理由は，異なるアプリケーションも暗号化によりポート番号が同じになってしまうためである．一例として，ポート番号 22 は Secure Shell(SSH)の Well-known ポートであるが，SSH はセキュアなログイン，ファイル転送，任意のアプリケーションのポート転送などに使われる．

同様に，動的なポートを使うストリーミングトラヒックもポート番号による識別は難しい．映像音声のストリーミングでは，各メディア（音声，映像）毎に動的に異なるポート番号を使ったトラヒックによる通信が行われることが多く，ポート番号からは識別できない．

2.1.2. ペイロード解析による識別

ペイロード解析とは，パケットのペイロードの内容を解析して識別を行う方法である．ペイロードは OSI 参照モデルの 7 階層のうち，アプリケーション層に属し，各アプリケーション毎に形式が決まっている．あらかじめ知っているそれぞれのアプリケーションのペイロードのシグニチャー（パターン）と，パケットのペイロードと比較することで，どのアプリケーションのトラヒックか識別することができる．Wireshark[20]はペイロード解析を行い数百のアプリケーションの識別を行うことができ，ほぼ 100%のトラヒックの識別が可能である．

ペイロード解析の一例として，Intrusion Detection System (以下 IDS) と呼ばれるネットワーク装置がある．IDS は，ペイロード解析とシグニチャーのマッチングを行い，不正侵入を検知を行っている．

このようにペイロード解析による識別は，既知のアプリケーションのトラヒックについては，高い精度で識別できるが，ポート番号による識別より計算負荷が大きい，通信内容が暗号化されていると解析が困難といった欠点がある．また，ペイロードにはメールの本文などプライバシーに関する情報も含まれるため，プライバシー侵害も懸念される．

2.2. 関連研究

前説で述べたような，ポート番号やペイロードを参照せずにトラヒックを識別する

手法を提案する関連研究について，概要と考察を述べる．

2.2.1. ホストの挙動による識別

[3]によれば，ホストの挙動によりトラヒックを識別できる手法が提案されている．この手法では，監視している複数のホストのトランスポート層の情報を含まないネットワークレベルの情報のみを利用し，アプリケーション固有の挙動（シグニチャー）を見つけて，ホストの動作を分類することで，監視ネットワーク内でトラヒックを識別することができる．トランスポート層の情報を使わないため，暗号化されたトラヒックに対しても適用できる．しかし監視下のホストの挙動から識別するため，個々のフロー単位で識別は困難であり，トラヒックを識別する目的，用途によっては，この手法は適さない．

2.2.2. トラヒックフローの統計情報による識別

フロー単位で識別できる手法として，トラヒックフローの統計情報を利用した識別手法が提案されている．トラヒックフローを，通信の開始から終了までの一連のパケットの送受信と定義すると，そのトラヒックフロー毎に，様々な統計情報を計算することができる．例えば，統計情報として，総転送パケット数，総転送バイト数，平均パケットサイズなどが考えられる．この統計情報にアプリケーション毎に特徴があると仮定し，統計情報を元にアプリケーショントラヒックを識別する手法である．統計情報にはポート番号やペイロード自体を含まず，暗号化されていても特徴は抽出できると想定されるため，2.1 節であげた課題をクリアしている．

しかし，統計情報の特徴を識別するためには，あらかじめ多数のトラヒックフローのデータが必要となるという課題がある．また，統計情報を計算するためには，基本的には通信が終了している必要がある．通信の途中で統計情報を随時計算することも可能だが，識別のためには，通信開始からある程度のパケット数が必要という欠点がある．

この手法の関連研究は多数提案されている．これらの研究を，[4]がまとめたチャートをベースに，入力データ，手法，出力結果で整理した図が図 2.1 である．

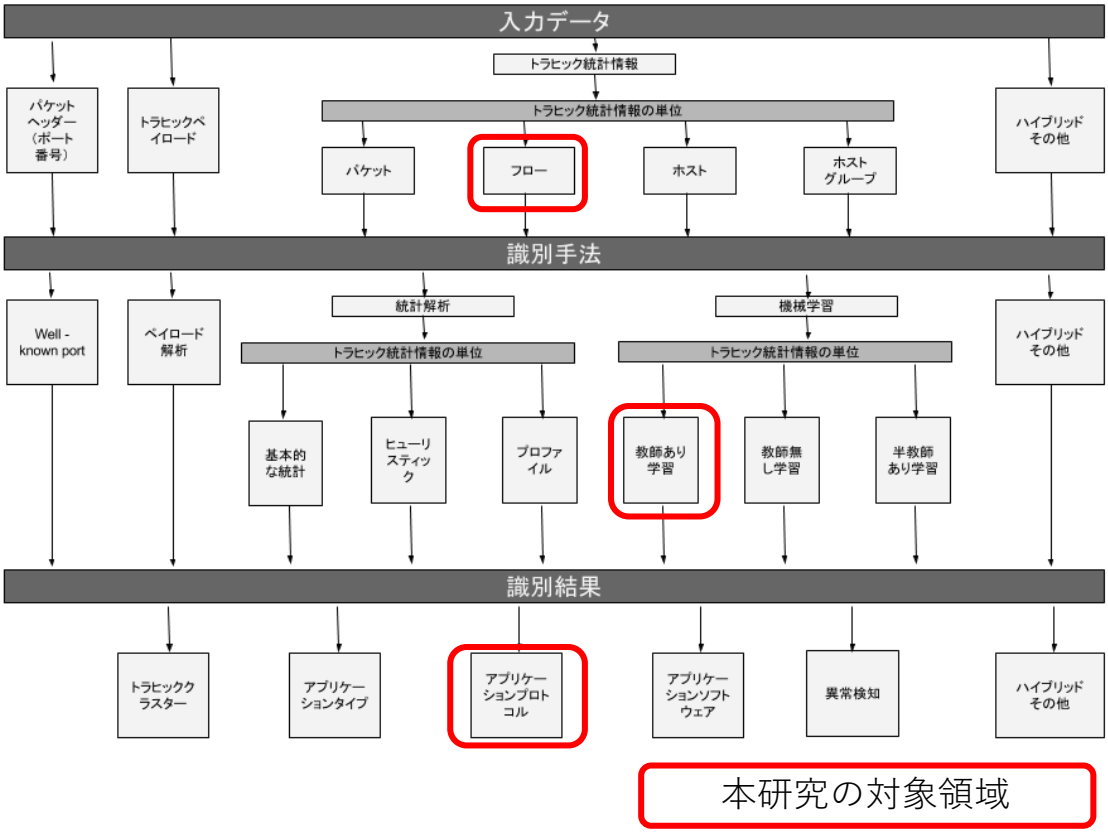

図 2.1 関連研究手法まとめ

本研究の提案手法に近い研究として，[5]は，Multi layer perceptron (以下 MLP) による識別を提案しており，C4.5, SVM, Bayes などの機械学習手法より高い精度を実現している．

2.2.3. 関連研究の考察

以上，本章で概観した関連研究の中で，トラヒックフローの統計情報を入力データとし，教師あり機械学習の手法を使った研究に着目した．これは近年，機械学習の中でディープラーニング（深層学習）手法が著しく進化し，画像の識別などの分野で高い精度を実現していることから，トラヒックフローの統計情報による識別でも，従来の機械学習による識別精度より高い精度を実現できる可能性があるのではないか，と考えたためである．

教師あり機械学習の手法を使った関連研究でディープラーニングの手法を使ったものとして，2.2.2 節で述べた論文[5]があり，実質的に本研究の提案手法とほぼ同じ

手法である．ただし，本研究と比較して，入力データに利用するデータセットの統計情報，出力の識別がアプリケーションのタイプであり大まかな分類である事，Voip などストリーミングトラヒックのデータがない事，が異なっている．

本研究では，これまで関連研究の無い，ディープラーニングの手法を使った Voip ストリーミングトラヒックフローの統計情報による識別において，機械学習手法より高い精度を実現できると仮説を立て，その手法を提案し，実験，評価を行った．

3. 提案する識別手法

本章では，トラヒックフローの統計情報による識別において，より高い精度を実現するための手法として，全結合の Deep Neural Network によるディープラーニングを利用することを提案する．まず入力データとして利用するデータセットについて述べる．次に全結合の Deep Neural Network の構成を示し，精度を改善する様々な手法について述べる．

3.1. データセット

本研究の実験では，NIMS, NIMS2 （以下両方合わせて NIMS）データセットを使用した[6][7][8][9]．既存のデータセットを使うことで，精度を関連研究と比較することが可能となる．この NIMS データセットは，表 3.1 で示される暗号化・非暗号化のトラヒック，Voip(Voice over IP)のストリーミングトラヒックが含まれる[10][11].

表 3.1 : NIMS データセット

	トラヒック	フロー数
1	TELNET	1251
2	FTP	1728
3	HTTP	11904
4	DNS	38016
5	lime(P2P)	646271
6	ssh(localForwarding)	2557
7	ssh(remoteFowarding)	2422
8	scp	2444
9	sftp	2412
10	x11	2355
11	shell	2491
12	Primus	9591
13	Zfone	29961

ニューラルネットワークの学習をバランスよく行うため，フロー数を均等にするようランダムに 2000 フローずつ抽出した．ただし，2000 フローに満たない TELNET

と FTP については，1000 フローを抽出した．この結果，抽出された合計 24000 のフローを本研究の実験で使用するデータセットとした．

また，NIMS データセットは，一フロー毎に，表 3.2 で示される 22 の統計情報と，教師データとしてフローのアプリケーション情報が含まれている．

表 3.2：トラヒックフロー統計情報

	統計情報	名前
1	Protocol	proto
2	Duration of the flow	Duration
3	# Packets in forward direction	fpackets
4	# Bytes in forward direction	fbytes
5	# Packets in backward direction	bpackts
6	# Bytes in backward direction	bbytes
7	Min forward inter-arrival time	minfiat
8	Mean forward inter-arrival time	meanfiat
9	Max forward inter-arrival time	maxfiat
１０	Std deviation of forward inter-arrival times	stdfiat
１１	Min backward inter-arrival time	minbiat
１２	Mean backward inter-arrival time	meanbiat
１３	Max backward inter-arrival time	maxbiat
１４	Std deviation of backward inter-arrival times	stdbiat
１５	Min forward packet length	minfpkt
１６	Mean forward packet length	meanfpkt
１７	Max forward packet length	maxfpkt
１８	Std deviation of forward packet length	stdfpkt
１９	Min backward packet length	minbpkt
２０	Mean backward packet length	meanbpkt
２１	Max backward packet length	maxbpkt
２２	Std deviation of backward packet length	stdbpkt

NIMS データセットの統計情報にはポート番号やペイロードは含んでおらず，前章で述べたようにこのデータセットを使った識別は，ポート番号やペイロードに依存しない．

3.2. Deep Nueral Network (DNN)

本研究では，機械学習の一手法である全結合の Deep Neural Network（以下 DNN）を，トラヒックフローの統計情報による識別で，より高い精度を実現するための手法として利用した．DNN とは，ニューラルネットワーク（以下 NN）の隠れ層の数を深くした構成になっている．

NN とは，図 3.1 に示したように，パーセプトロンを一つのノードと考え，ノードを複数の層に組み合わせたネットワークである．全結合の NN の特徴として，同じ層の中のノードは関連が無く，層間は全てのノードが個別の重みがついた構成となっている．

NN による識別は，学習と判別の二つのフェーズがあり，学習フェーズでは，Training データを入力層から隠れ層，出力層の順にデータを重みをかけながら伝搬させ，出力されたデータと教師データの誤差を逆伝搬させることで，重みを更新し，誤差が収束するまで繰り返しモデルの Training を行う．一方判別フェーズでは，同様に Test データを入力層から隠れ層，出力層の順にデータを重みをかけながら伝搬させ，出力層の結果が判別結果となる．

この NN の層を深くすることで，特徴量の抽出を自動的に行うことができるようになるというメリットがある反面，誤差逆伝搬による重みの更新，すなわち，モデルの Training が収束しないという欠点があった．近年，効率的に Training を行うためのアルゴリズムとして，Pre-Training, Dropout など，複数提案されている．本研究では Dropout を採用し Training を行った．

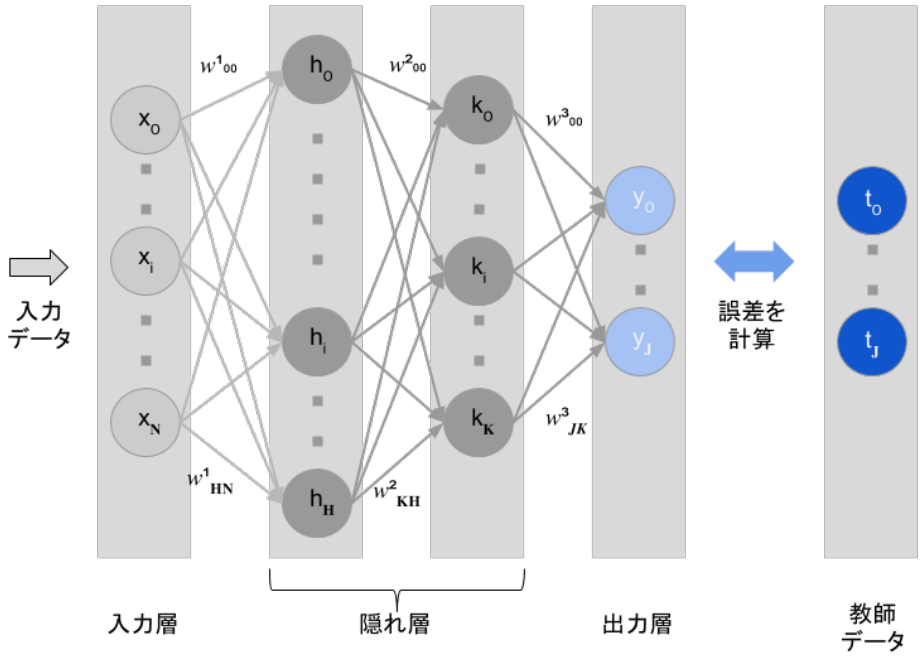

図 3.1 Neural Network

3.3. Dropout

本研究では，DNN モデルの過学習を避け汎化性能を向上させるため，隠れ層の一部で Dropout を適用した．Dropout は，図 3.2 に示したように，NN のモデル Training 時に，隠れ層のノードを一定の確率で無効にして Training を実施する手法である[12]
.

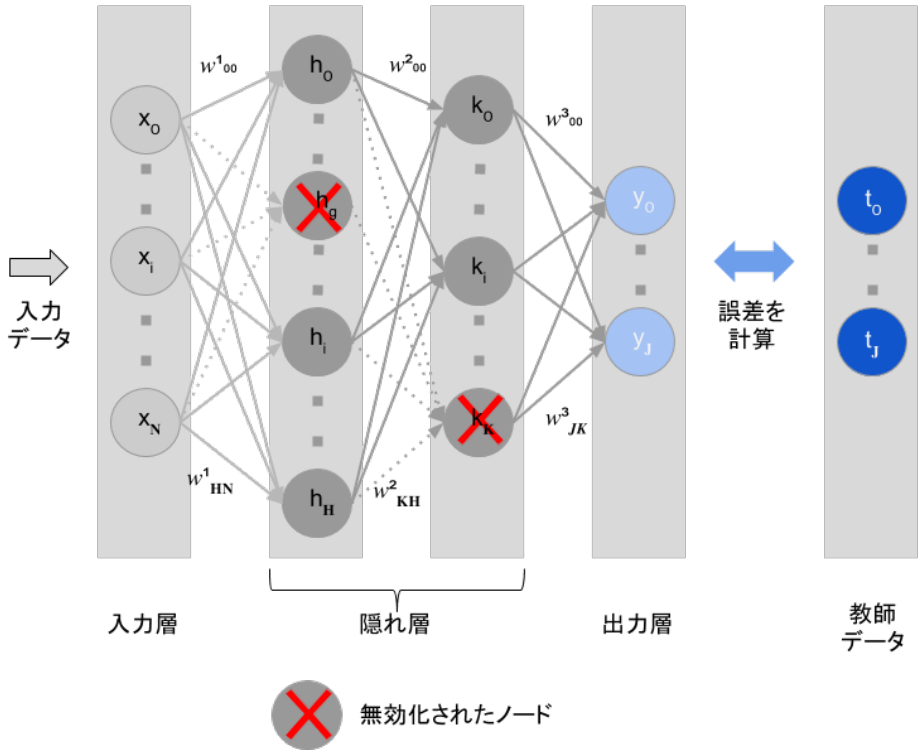

図 3.2 Dropout

Dropout により，隠れ層のノードを一定の確率で無効にすることで，多数の異なる構成のモデルの組み合わせが出来，機械学習の一手法であるアンサンブル学習のように，異なるモデルの組み合わせの平均を取ることで，汎化性能が向上する．

3.4. Cross-validation (K-fold cross-validation)

本研究では，DNN モデルの精度評価のために，K-fold cross-validation を行った．K-fold cross-validation は，モデルの入力データとなる，正解ラベルのついたデータ数が少ない時に，精度評価に有効な方法である[13]．本研究の実験では，図 3.3 のように，4 分割の一つを Validation データ，残り 3 つを Training データとし，テストデータを入れ替えながら，4 Round の各 Round 毎に Validation データセットの精度評価を行い，4 Round の精度の平均をそのモデルの評価精度とした．

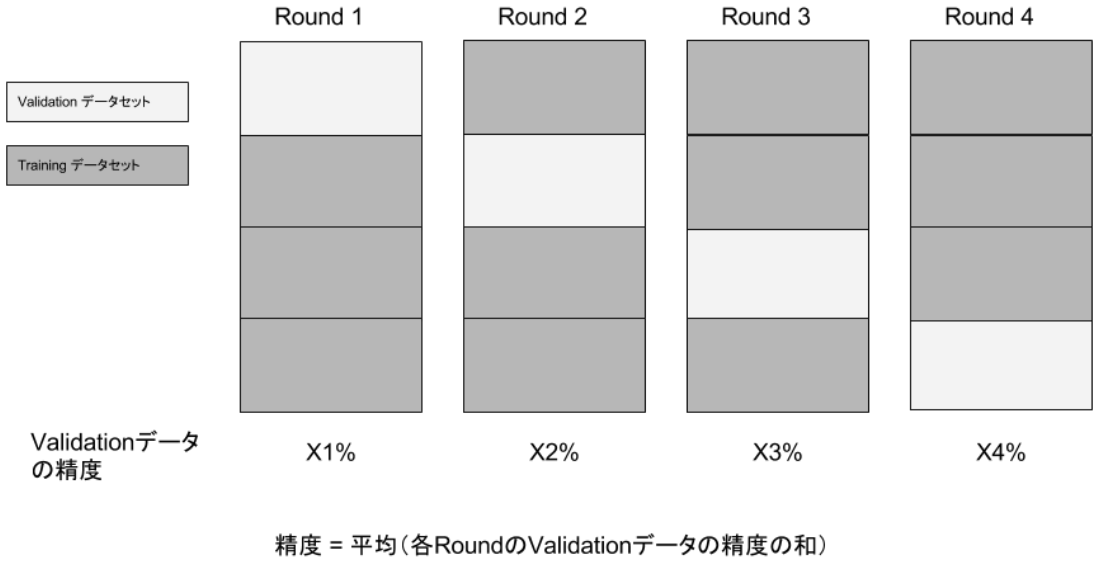

図 3.3 K-fold cross-validation

3.5. Hyper-Parameter 探索

本研究では，DNN モデルの精度改善ために，Hyper-Parameter 探索を行った．Hyper-Parameter は，ニューラルネットワーク自体のパラメータで，活性化関数や最適化アルゴリズム，隠れ層のユニット数など，多様な選択肢から最適な組み合わせが存在する．本研究の実験では，自動的にもっとも精度の良い Hyper-Parameter の組み合わせを探索した．探索方法には，全ての組み合わせを網羅的に試す Grid-Search と，ランダムに Hyper-Parameter の組み合わせを選択し探索する Randomized-search の，二つの手法がある[14]．本研究では，探索時間の制約から，Grid-Search による全探索は実施できなかったため，Randomized-search で最適な組み合わせの方向性を徐々に絞り込んでいく方針を採用した．

Randomized-search では，全データセットの 10%（2400 フロー）をテストデータとして，分離し，残り 22600 フローを使って 4 分割の Cross Validation を行った．つまり 25%（5650 フロー）を Validation データ，残り（16750 フロー）を Training データとして使用し，4 回の精度を平均し，そのパラメータセットの精度とした．

表 3.3 に本研究で探索したパラメータを列挙する．

表 3.3：探索したパラメータ

	パラメータ	探索範囲
1	隠れ層 1,2 のノード数	[150, 200, 250, 300]
2	隠れ層 3 のノード数	[80, 90, 100, 120]
3	隠れ層 4 のノード数	[40, 45, 50, 60]
4	隠れ層 5 のノード数	[40, 50, 60, 70]
5	隠れ層 6 のノード数	[40, 50, 60, 70]
6	ミニバッチサイズ	[100, 200, 500]
7	初期値のアルゴリズム	["lecun_uniform", "glorot_normal", "glorot_uniform", "he_normal", "he_uniform"]
8	optimizer	["RMSprop", "Adagrad", "Adam", "Nadam"]
9	activation	["relu", "tanh", "softsign"]

3.6. ツール

DNN モデルの作成，学習，判別のために使ったツールとして，TensorFlow をバックエンドに使った Keras ニューラルネットワークライブラリを利用した[15][16]. TensorFlow は機械学習の中でも特にディープラーニングに特化した, 学習と推論を行うことができるフレームワークである．処理に複数の GPU を使えることにも対応している．一方，Keras は，python 言語で記述された，高水準でより抽象モデルの記載が可能な，ニューラルネットワークライブラリである．実際の学習や推論などの処理は，TensorFlow などのフレームワークをバックエンドとして使用する．Keras によりニューラルネットワークを使った実験を迅速に行うことが可能となる．

4. 実験結果と評価

本章では，前章で述べた提案手法である，DNN のモデルと Hyper-Parameter 探索の実験を行った機材, 環境と実験結果について述べ，同じデータセットを使った既存の論文と比較し評価する.

4.1. 実験機材と環境

本節では，実験に用いた機材と環境について述べる.

4.1.1. ハードウェア機材

実験に用いたハードウェア機材は下記の通りである.

- ✓ MacBook Pro 2012
- ✓ CPU：2.6GHz Core i7
- ✓ メモリ：16GB

4.1.2. ソフトウェア環境

実験に用いたソフトウェア環境は下記の通りである.

- ✓ OS：macOS Sierra
- ✓ 言語：Python　3.5.2
- ✓ フレームワーク：TensorFlow 1.0.0 / Keras 1.2.2
- ✓ ライブラリ：numpy 1.12.0 / scikit-learn 0.18.1
- ✓ 開発環境：jupyter notebook　1.0.0

4.2. 識別結果

本節では，識別結果を評価する指標，パラメータ探索の結果決定したパラメータ，決定したパラメータによる識別結果，汎化性能，識別状況の可視化，識別に有効な統計情報について述べる.

4.2.1. 指標

識別結果の評価を，予測結果の評価指標の一つである F 値（F-measure）と AUC で

行う.

F 値は適合率（precision）と再現率（recall）の調和平均で定義される指標であり，トレードオフの関係にある適合率と再現率を総合的に評価する指標であり，0 から 1 の値をとる.1 に近いほど,適合率と再現率のバランスよく高性能であることを示す.

F-measure = (2*precision*recall) / (precision+recall)

ここで，適合率は識別された結果の数（N）のうち正しく識別出来た数（R）の比率（R/N）であり，再現率は Test データにある真のアプリケーショントラヒックのフロー数（C）のうち，正しく識別出来た数（R）の比率（R/C）を意味する.

AUC は area under the curve の略で，ROC 曲線の下側の面積を指し，面積が大きいほど，その値が 1 に近く[19].

4.2.2. パラメータ探索結果

本研究で行った Hyper-Parameter 探索の結果，表 4.1 のパラメータの組み合わせが F 値が 0.990 で最も高かった．Cross Validation の Round ごとの精度の分散も非常に小さいことから(std: 0.001)，安定して高い精度のパラメータの組み合わせであると言える.

表 4.1：決定したパラメータ値

	パラメータ	値
1	隠れ層 1,2 のノード数	200
2	隠れ層 3 のノード数	80
3	隠れ層 4 のノード数	50
4	隠れ層 5 のノード数	50
5	隠れ層 6 のノード数	50
6	ミニバッチサイズ	50
7	初期値のアルゴリズム	glorot_uniform
8	optimizer	Nadam
9	activation	relu

上記のパラメータの組み合わせから，最終的に決定したニューラルネットワーク

の構成は図 4.1 の通りである.

Layer (type)	Output Shape	Param #	Connected to
dense1 (Dense)	(None, 200)	4600	dense_input_202[0][0]
relu1 (Activation)	(None, 200)	0	dense1[0][0]
dropout1 (Dropout)	(None, 200)	0	relu1[0][0]
dense2 (Dense)	(None, 200)	40200	dropout1[0][0]
relu2 (Activation)	(None, 200)	0	dense2[0][0]
dropout2 (Dropout)	(None, 200)	0	relu2[0][0]
dense3 (Dense)	(None, 80)	16080	dropout2[0][0]
relu3 (Activation)	(None, 80)	0	dense3[0][0]
dense4 (Dense)	(None, 50)	4050	relu3[0][0]
relu4 (Activation)	(None, 50)	0	dense4[0][0]
dropout4 (Dropout)	(None, 50)	0	relu4[0][0]
dense5 (Dense)	(None, 50)	2550	dropout4[0][0]
relu5 (Activation)	(None, 50)	0	dense5[0][0]
dense6 (Dense)	(None, 50)	2550	relu5[0][0]
relu6 (Activation)	(None, 50)	0	dense6[0][0]
dense7 (Dense)	(None, 13)	663	relu6[0][0]
softmax7 (Activation)	(None, 13)	0	dense7[0][0]

Total params: 70,693
Trainable params: 70,693
Non-trainable params: 0

図 4.1 決定した Neural Network 構成

4.2.3. 識別結果

前節のパラメータ探索結果でベストのパラメータの組み合わせでトレーニングさせたモデルによる，各アプリケーションのトラヒックフロー毎の評価結果は表 4.2 に示す.

表 4.2：識別結果

	トラヒック	precision	recall	F 値	support	AUC
1	TELNET	1.00	1.00	1.00	100	1.000
2	FTP	1.00	1.00	1.00	120	1.000
3	HTTP	0.98	0.99	0.99	192	0.994
4	DNS	0.98	0.98	0.98	202	0.996
5	lime(P2P)	0.97	0.96	0.96	182	0.985
6	ssh(localForwarding)	1.00	1.00	1.00	225	1.000
7	ssh(remoteFowarding)	1.00	1.00	1.00	208	1.000
8	scp	1.00	1.00	1.00	191	1.000
9	sftp	1.00	1.00	1.00	202	1.000
１０	x11	1.00	0.99	0.99	191	0.999
１１	shell	0.99	1.00	0.99	193	1.000
１２	Primus	0.99	0.98	0.99	173	0.997
１３	Zfone	0.98	0.99	0.98	221	0.996
	平均（support のみ合計）	0.99	0.99	0.99	2400	0.996

この結果から，提案手法は暗号化・非暗号化のトラヒック，Voip(Voice over IP)のストリーミングトラヒックのいずれの識別においても，高い精度で識別できていることが確認できる．

4.2.4. 汎化性能

4.1.2 節で決定したパラメータの組み合わせによるモデルが偶然高い精度だったわけではなく，パラメータの汎化性能が統計的に有意であることを確認するため，パラメータの組み合わせを 4.1.2 節の通り固定し，50 回の Training により 50 のモデルを生成し，各モデルの識別結果の分散が小さく安定して高い精度であることを確認した．その際，各 Training 毎に，データセット全体を Validation データ，Training データ, Test データにランダムに分割しデータを使用し，データに依存がおきないよう実験を行った．50 回実施した結果を Precision（図 4.2）, Recall（図 4.3）, F1-score（図 4.4），Support（図 4.5）に示す．ここで Support はテストデータの項目数である．

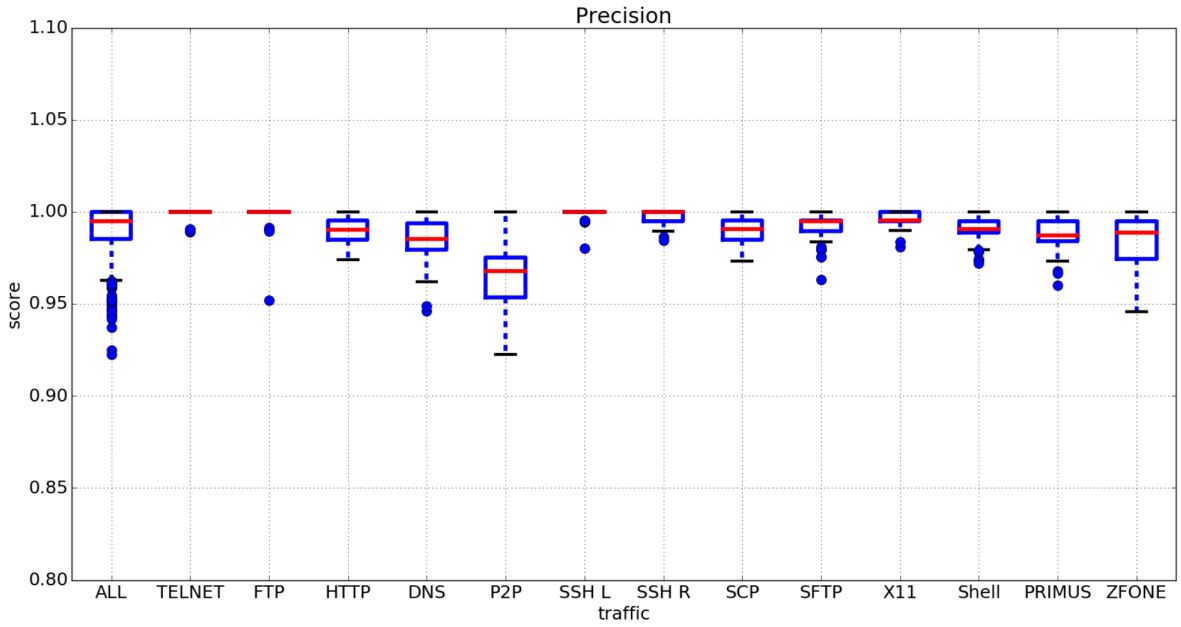

図 4.2 汎化性能(Precision)

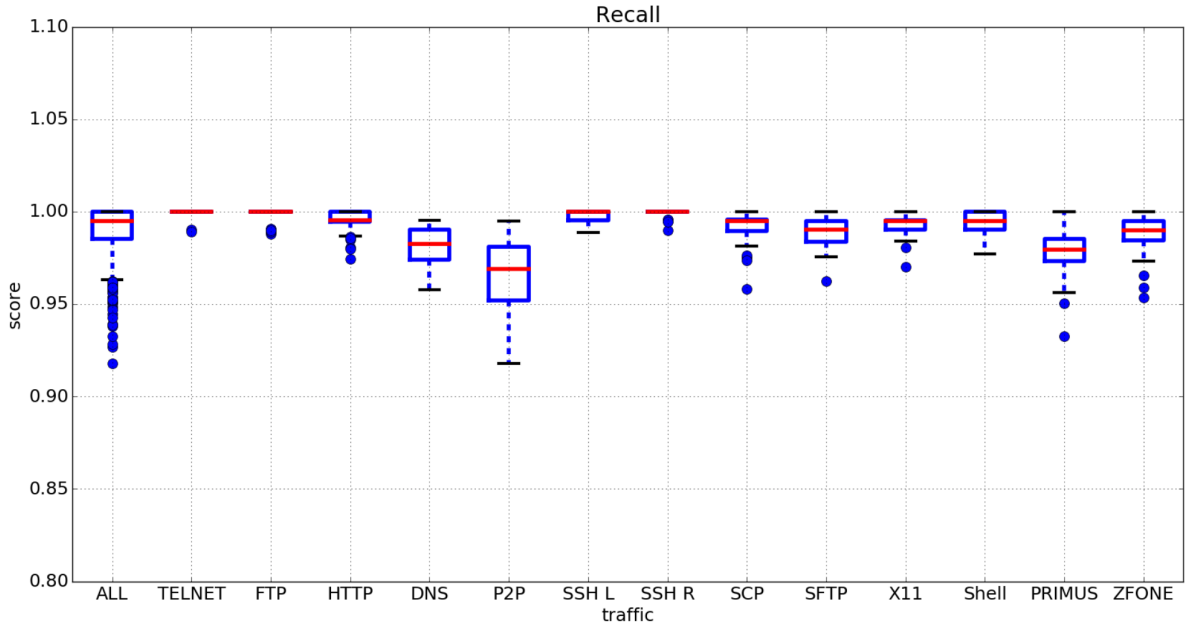

図 4.3 汎化性能(Recall)

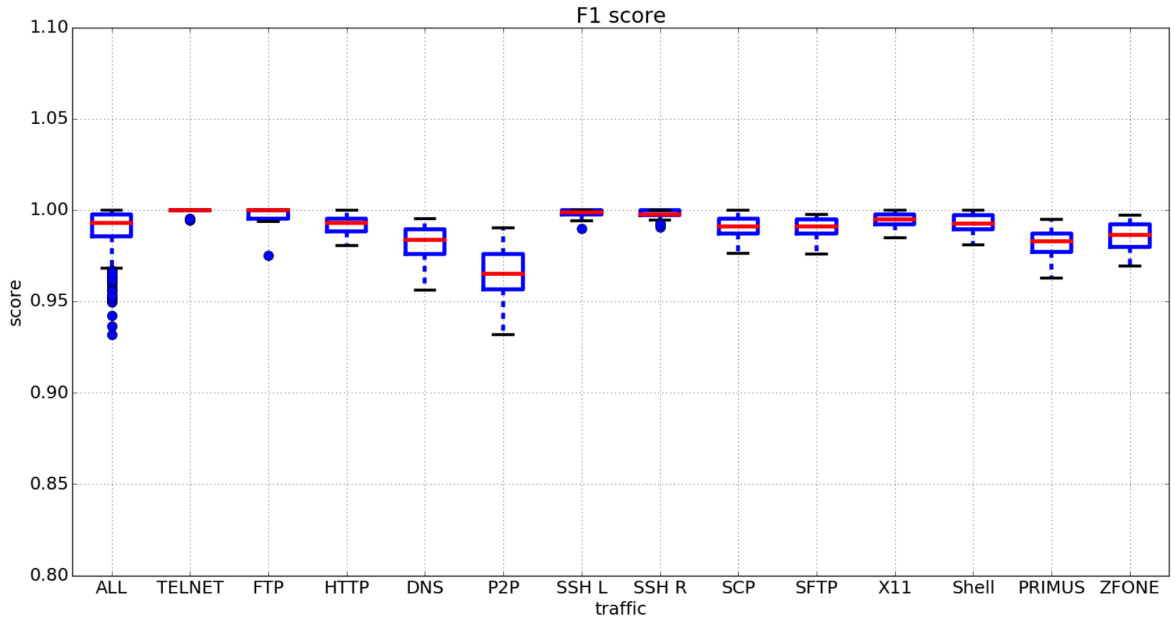

図 4.4 汎化性能(F1-score)

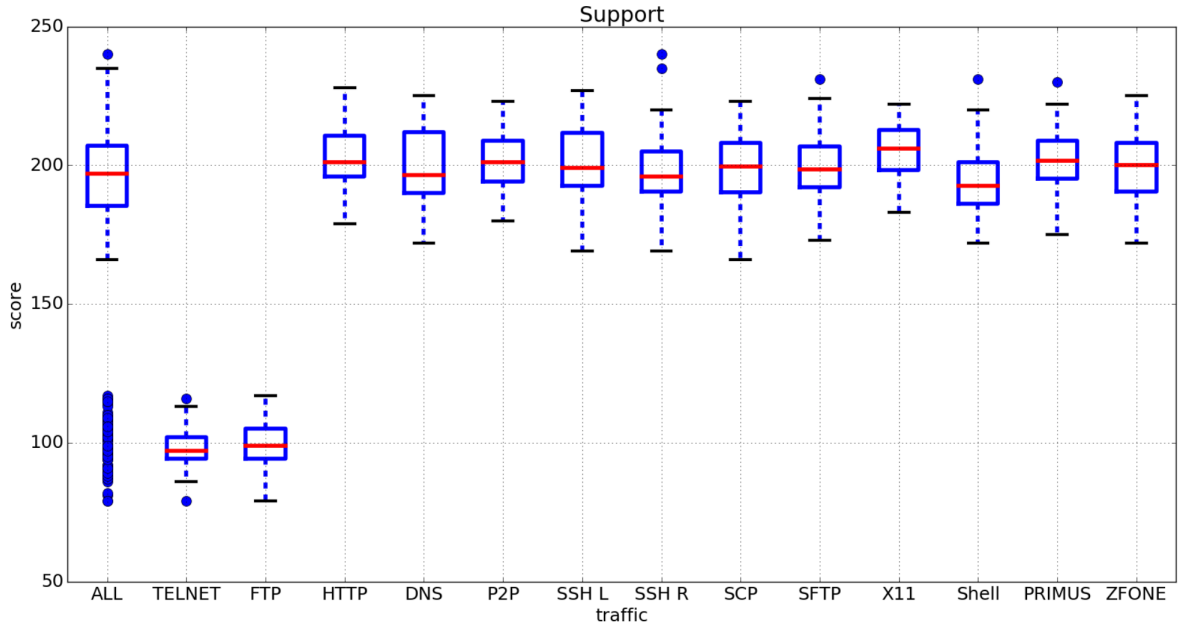

図 4.5 汎化性能(Support)

以上の結果から，提案手法と決定したパラメータは，データに依存することなく，Precision, Recall, F1-score いずれも高い精度を実現できていることが確認できた．

4.2.5. 識別の可視化

一般的に，ニューラルネットワークによる学習は，どのように隠れ層が学習されているかわかりにくい．そこで識別の様子を可視化するため，第 6 層のノード数を 2 にしてその各ノードの出力値を両対数の 2 次元にプロットするし，隠れ層を散布図（図 4.6 図 4.7 図 4.8）に表現した．x, y 軸とも DNN の内部表現の値であるため，単位は無い．ほとんどのフローはアプリケーショントラヒックごとにクラスターに分離されており，DNN により特徴量を自動的に抽出できていることが確認できる．

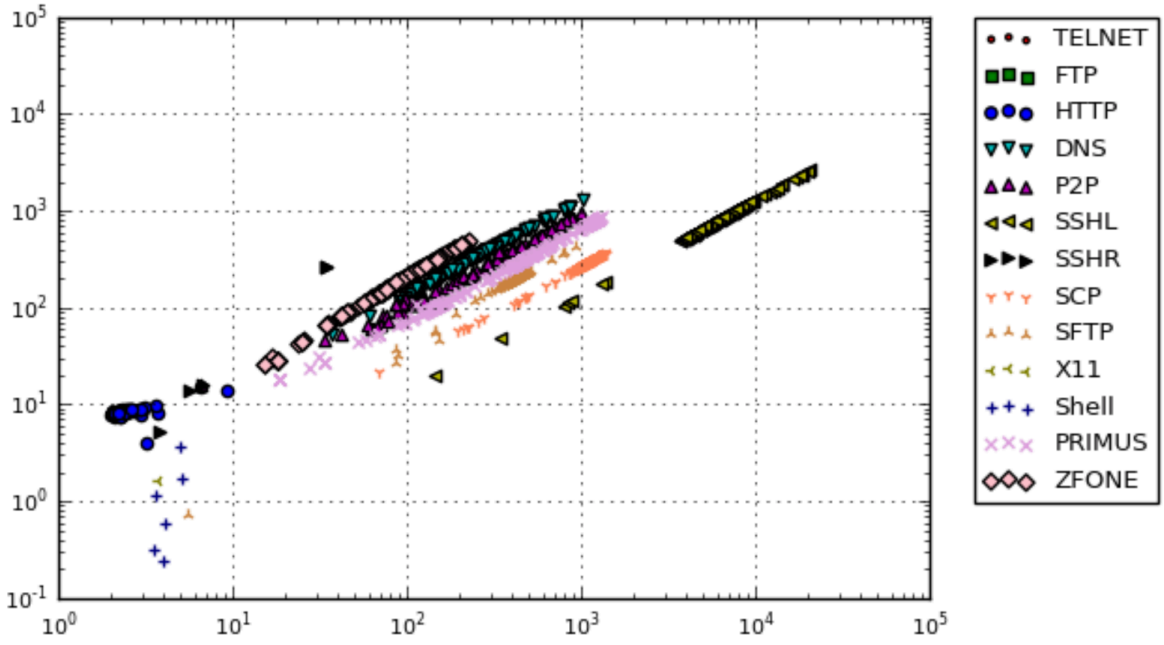

図 4.6 全フロー散布図

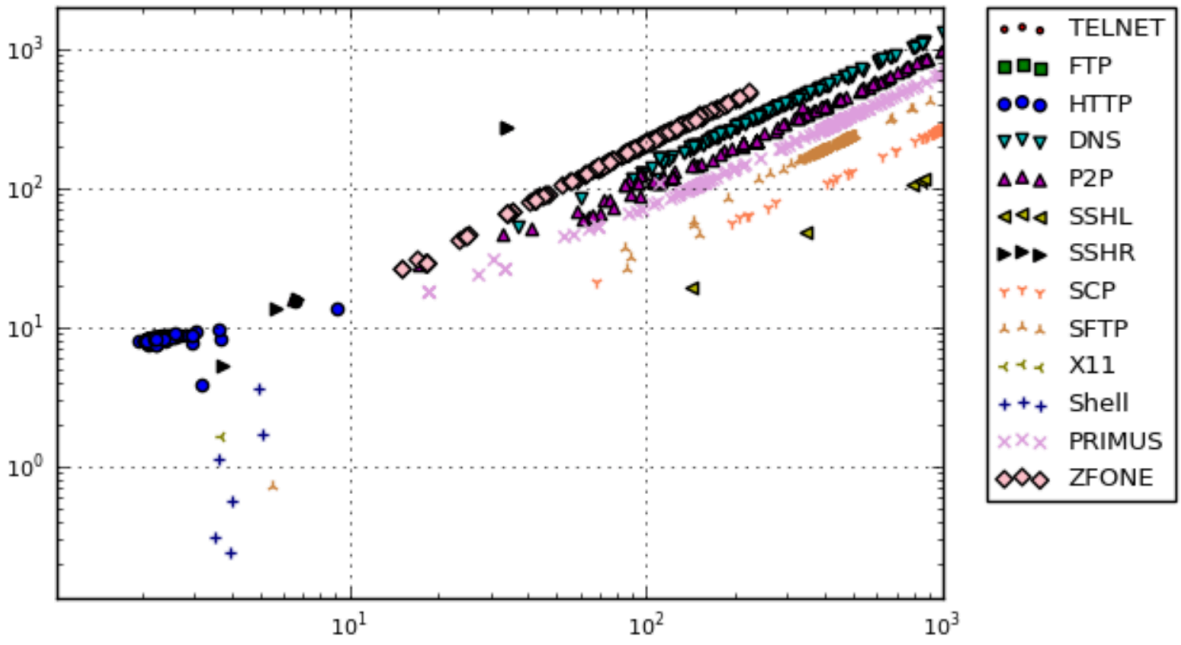

図 4.7 原点付近を拡大

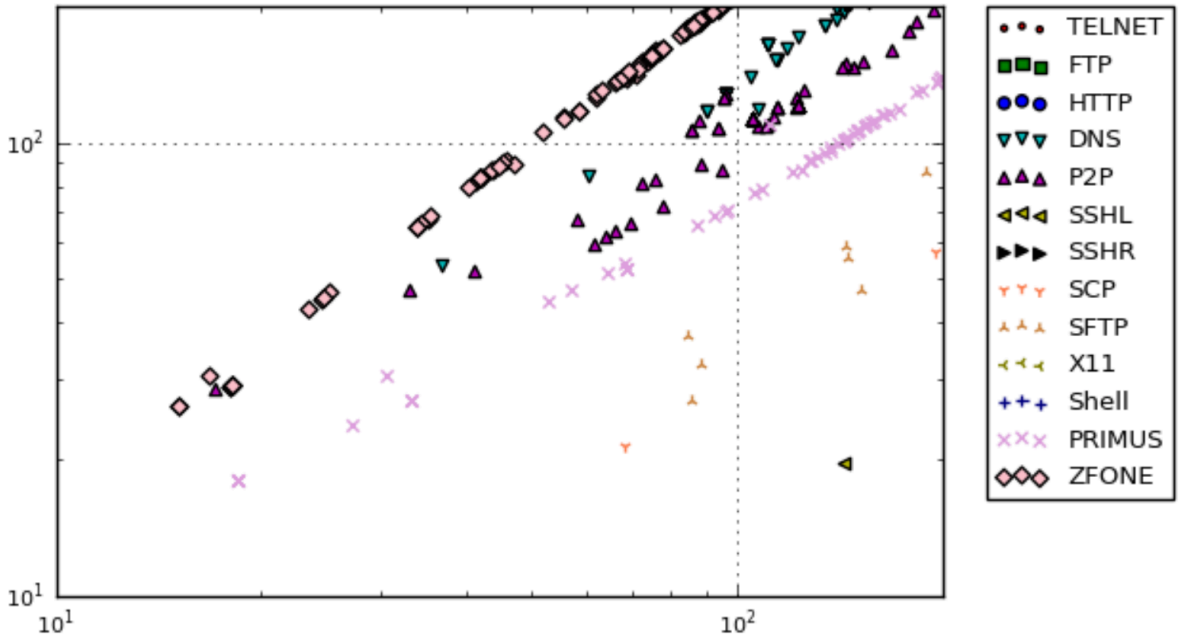

図 4.8 原点付近をさらに拡大

4.2.6. 識別に有効な統計情報

本研究では，入力データとして 22 のトラヒックフローの統計情報を用いているが，どの情報が識別により重要度の高い情報かを確認するため，隠れ層第一層の入力の重みの二乗平均を計算した結果を表 4.3 と図 4.9 に示す.

表 4.3：識別に有効な統計情報

順位	フロー統計情報	重み
1	total_fpackets	6.49139261
2	total_fvolume	5.6524272
3	duration	3.58002257
4	min_fiat	3.34661794
5	std_fpktl	2.35084152
6	min_biat	1.72449863
7	min_fpktl	1.41581297
8	max_fpktl	1.02357852
9	mean_fpktl	0.98799551
10	total_bvolume	0.96912932
11	total_bpackets	0.90878588
12	std_bpktl	0.84534395
13	mean_biat	0.44610718
14	mean_fiat	0.43567511
15	std_fiat	0.37644351
16	std_biat	0.3721751
17	max_biat	0.35835418
18	mean_bpktl	0.23130643
19	min_bpktl	0.20463151
20	max_fiat	0.17666137
21	proto	0.12209202
22	max_bpktl	0.11297533

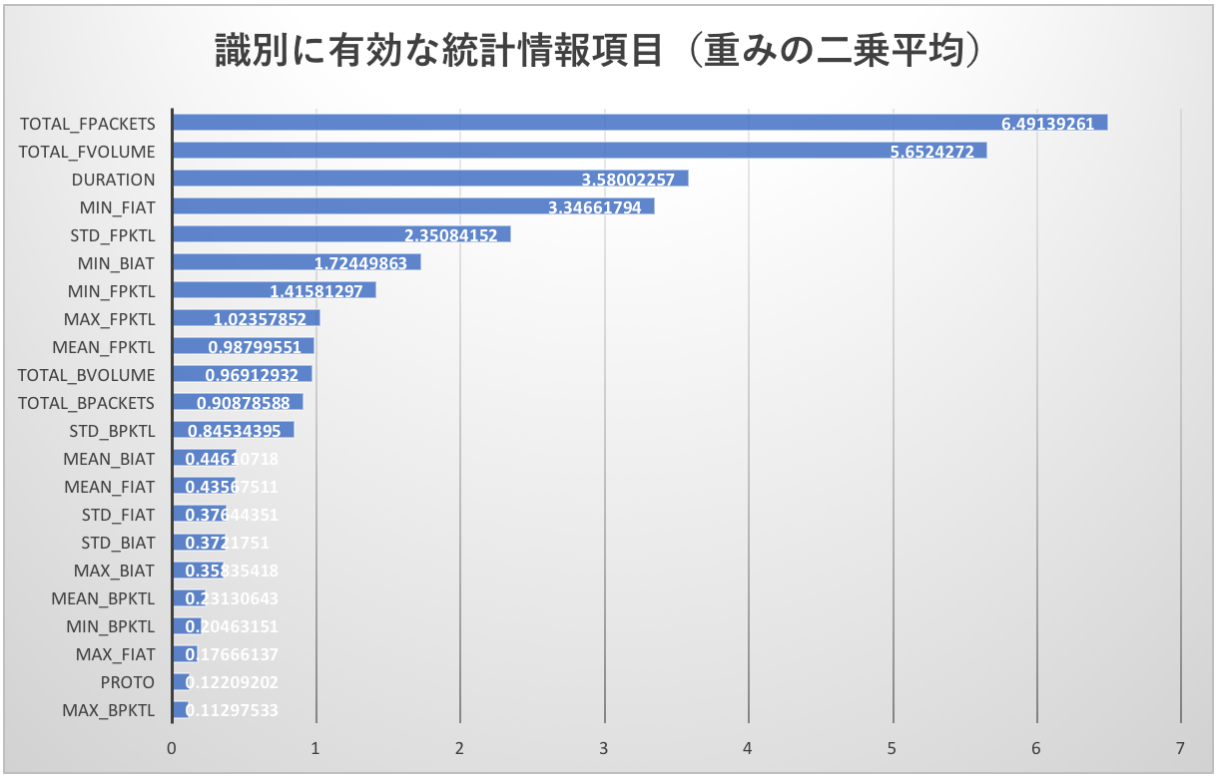

図 4.9 識別に有効な統計情報

この結果から，total_fpackets, total_fvolume, duration のように，フロー全体を表す統計情報の重要度が高いと言える．

4.3. 既存論文との比較

本節では，同じデータセットを使った，関連研究の結果と比較し評価する．[6]によれば，C4.5 と呼ばれる機械学習の一つの手法である決定木を使った識別手法が提案されている．この論文では評価指標として，DR と FPR が用いられている．DR は検出率を意味し前節の再現率と同じ指標で，FPR は誤検出率を意味する．同じ指標で比較した結果を表 4.4 に示す．

表 4.4：C4.5 との比較

		提案手法（DNN）		C4.5	
	トラヒック	DR	FPR	DR	FPR
1	TELNET	1.00	0.0	0.998	0.0
2	FTP	1.00	0.0	0.995	0.0
3	HTTP	0.99	0.002	0.991	0.0
4	DNS	0.98	0.002	1.0	0.0
5	lime(P2P)	0.96	0.002	0.995	0.001
6	ssh(localForwarding)	1.00	0.0	1.00	0.0
7	ssh(remoteFowarding)	1.00	0.0	1.00	0.0
8	scp	1.00	0.0	0.828	0.006
9	sftp	1.00	0.0	0.967	0.264
１０	x11	0.99	0.0	0.714	0.033
１１	shell	1.00	0.0	0.997	0.0

（参考）本研究の実験の FPR は，既知の下記の関係式から計算した.

(1) Recall = TP / (TP + FN)

(2) Rrecision = TP / (TP + FP)

(3) Support = TP / FN

(4) TP + FP + TN + FN = TS

(5) FPR = FP / FP + TN

(1)(2)(3)(4)より FP, TN を導き，(5)で FPR 計算した.

TP, FP, TN, FN, TS はそれぞれ True Positive, False Positive, True Negative, False Negative, Total Support で全テストフロー数である.

表 4.4 をグラフ化した図 4.10，図 4.11 を示す.

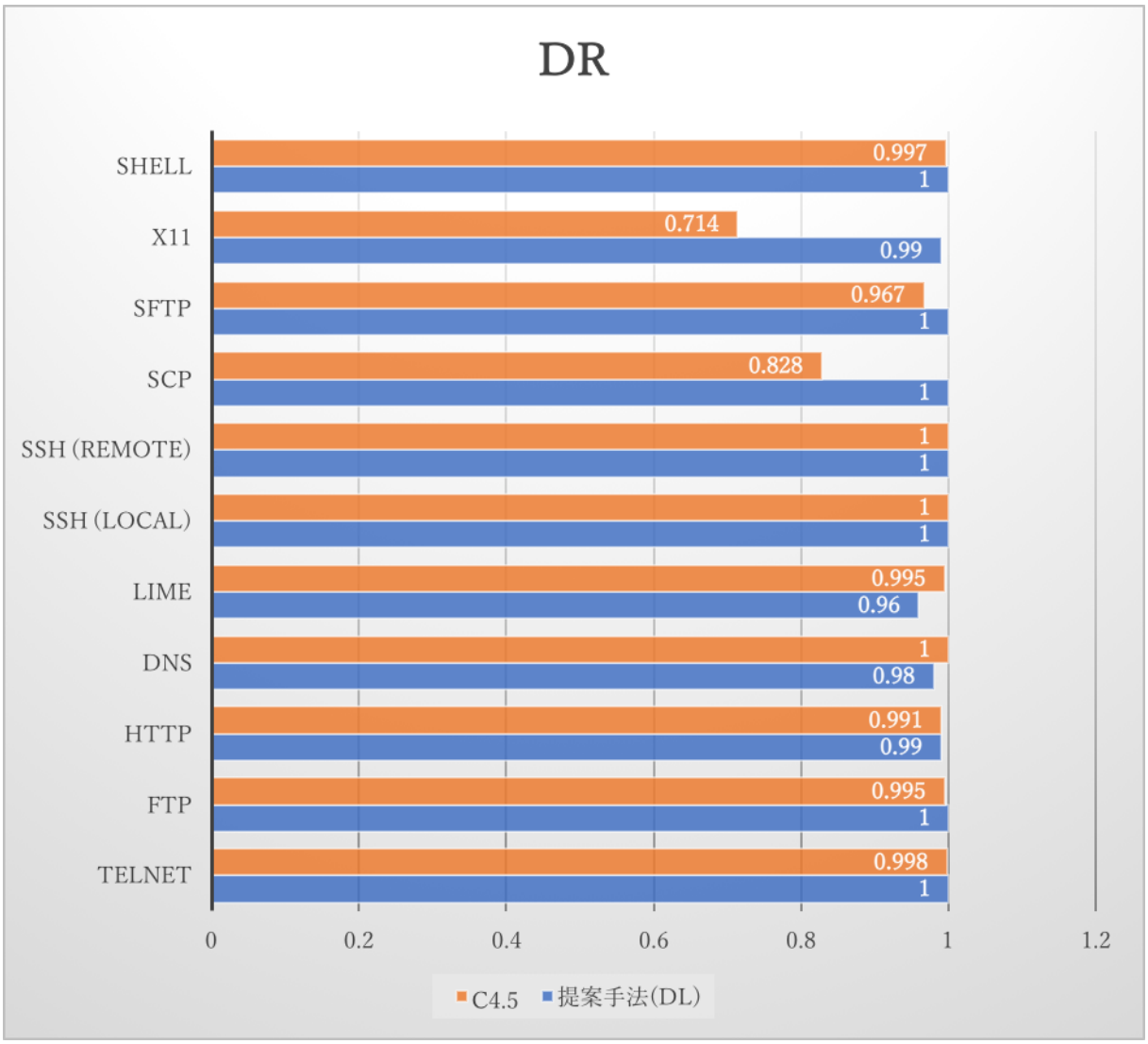

図 4.10 検出率（DR）の比較

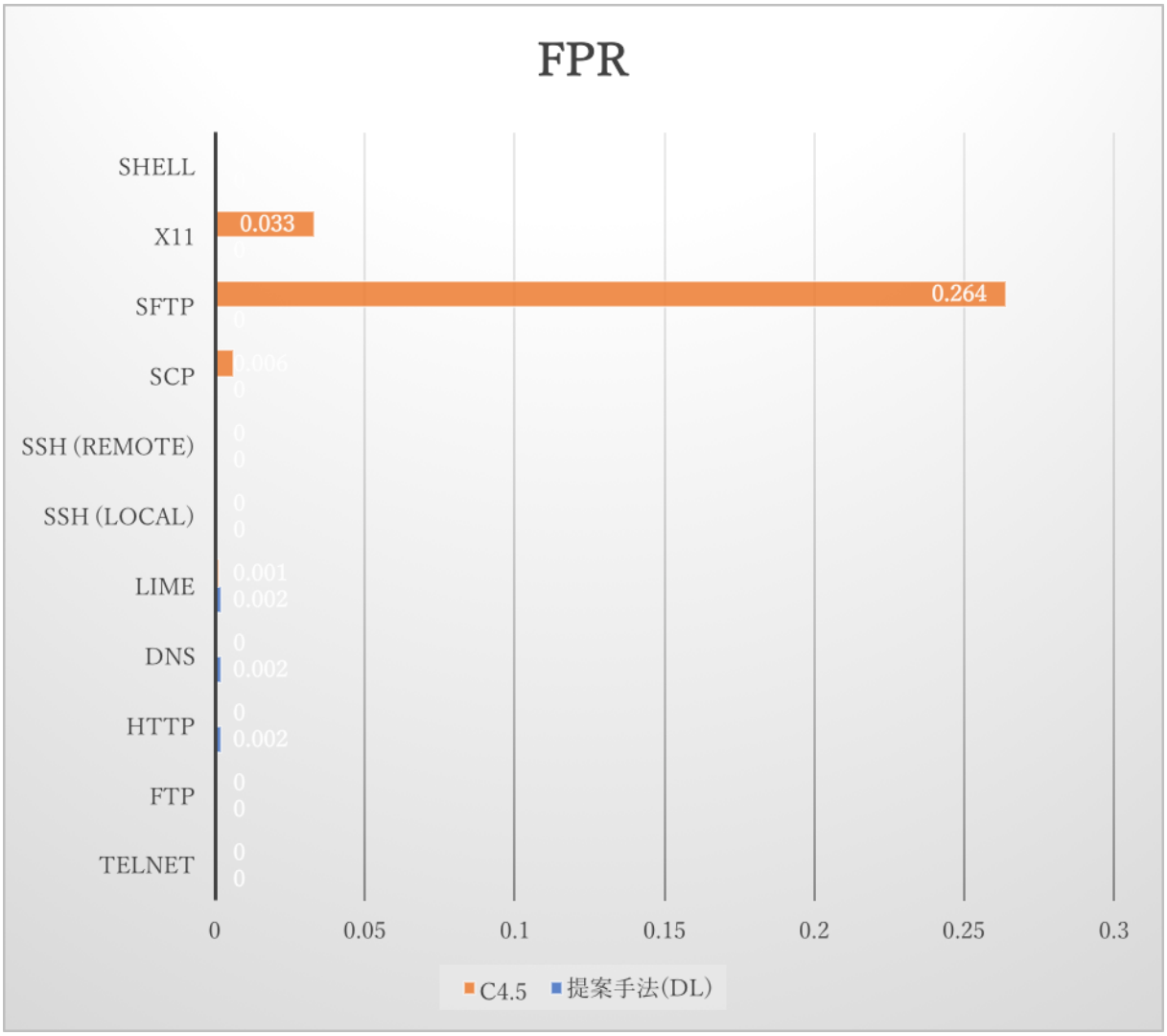

図 4.11 誤検出率（FPR）の比較

4.4. 処理性能

本研究の提案手法により実時間で識別が行えるか，考察する．

4.1 節で記載した実験環境において，データセットを元に計算すると，毎秒 8.8Mbit のトラヒックの識別を実時間で行える（詳細な計算は本節の最後で記載）．現実のネットワークトラヒックと比較すると識別性能が不足しているが，DNN の推論処理の特性から，下記の性能向上策が考えられる．

✓ GPU を用いることで数十倍の処理性能が得られる

✓ データ単位で並列実行可能であり，CPU や GPU を複数用いることで，比例した性能向上が得られる

仮に GPU により 100 倍，並列実行を 10 とすると，実験環境の 1000 倍の性能となり，8.8Gbit のトラヒック識別を実時間で行えることになる．例えば小規模なルータ装置[22]で処理できるトラヒックはギガビットイーサネットが 10 ポート程度であることから，中小規模の拠点でのトラヒックは実時間で識別できると推測できる．大規模な組織のバックボーンネットワークや，インターネットサービスプロバイダー（ISP），インターネットエクスチェンジ（IXP）など，大量のトラヒックを処理する必要のあるポイントに適用するには性能が不足しており，識別処理の効率化によって実時間処理性能の向上が必要である．

（参考）識別性能の計算

(1) DNN モデルの学習，推論に使用したデータセットの全 24000 フローから，平均の Duration と 1 フローのデータサイズを求め，一秒あたりの 1 フローの平均トラヒック量を求める．AvgTraffic = Avg(Data Size per flow) / Avg(Flow Duration)

(2) 実験環境において実測した推論時間（2400 フロー当たり 0.374 秒）から 1 フローの推論時間 (Inference Time = 0.374 / 2400)を求め，一秒当たり識別できるフロー数を求める（NumOfFlow=1/Inference Time）．

(3) 上記で計算した一秒あたりの平均トラヒック量（AvgTraffic）と識別できるフロー数（NumOfFlow）をかけて，識別性能を推測した（Inference Performance = NumOfFlow * AvgTraffic）．

4.5. Discussion

実時間処理の性能についても 4.4 節で考察したが，現実のインターネットのトラヒックに適用するには，性能が不足していることがわかった．DNN のモデルを，識別精度を落とさずに，より効率の良いモデルに改良していくことが必要である．その方法として次の対策が考えられる．

一つ目の対策は，隠れ層のレイヤー数，ノード数が多いほど，学習，推論時の計算量が増えるため，レイヤー数，ノード数を減らすことで，効率を改善できる．

もう一つの対策は，4.2.6 節に記載したように，識別に有効な統計情報のうち，友好度が少ない情報のインプットを削除することで，同様に効率の改善が期待できる．22 個の入力値のうち，どこまで削減できるかは，今後の研究課題である．

別の研究課題として，汎化性能のより高めるためには，より広範囲なネットワークのデータセットを利用する必要がある．3.1 節で述べたように，本研究で利用したデータセットは，関連研究の論文で使われたデータセットで，ある特定の日時の大学のトラヒックをキャプチャしたものであり，現在のインターネット上で使われる

プロトコルやトラヒックのパターンなど，利用状況は一致していない．

逆に Voip ストリーミングアプリケーションに限定し，類似するフローを正しく識別できるかに特化して精度を高めることも今後の研究テーマである．例えば，Skype[17], Google Talk/Hangout[18]その他主要なアプリケーションの識別は，C4.5 や SVM などの機械学習による識別の関連研究[9]はあるが，ディープラーニング手法を使ってより精度を高められるか，今後研究が必要である．

もう一つの研究課題として，オンライン処理の実現方法の検討がある．具体的には，本研究の入力情報はトラヒックフローの統計情報であるため，フローが終了してから統計情報を計算する必要があるが，トラヒックのフローの途中でリアルタイムに識別できる手法が求められる．例えばセキュリティーの観点からは，フローが終了してから識別するだけなく，フローの途中，できるだけ早い段階で識別し，不適切なトラヒックならば遮断するなどの対応をとる，といったニーズがある．リアルタイムに識別するためには，フローの途中で統計情報を随時計算し，DNN により分類することで実現できる可能性があり，さらなる研究が必要である．

5. 結論

本研究の提案手法により，従来手法より同等以上の精度で，IP ネットワーク上のトラヒック識別を実現することを確認できた．通常，precision と Recall, DR と FPR の評価指標はトレードオフの関係にあるが，提案手法はいずれも高い精度でバランスを保っている事が，F 値，AUC から確認できた．また，提案手法は，ポート番号やペイロードを参照せずにトラヒックを識別できることから，本研究で対象としなかったアプリケーションフローについても，暗号化・非暗号化・ストリーミングに関わらず，提案手法が識別に有効である可能性が高いと言える．

以下，論文のまとめを行う．

IP ネットワーク上のアプリケーションのトラヒックを正確に識別することは，帯域の管理やアプリケーションの QoS の確保，セキュリティの確保の点で，近年重要性が高まっている.しかし，トラヒックが暗号化されることで，識別が困難である，という課題が大きくなってきている．そこで本論文では，トラヒックのパケットサイズや到着間隔などの統計情報から抽出した特徴量に基づいて識別する方法に，DNN を適用して従来よりもより精度を高めることを目的とした研究を行った．

第 1 章では，トラヒック識別の必要性と，課題について述べ，DNN を適用し，高い精度と汎化性能を実現する検討, 提案がすることが本論文の目的であることを示した．

第 2 章では，関連研究について述べ，本論文との違いを明確化した．

既存手法の一つであるポート番号による識別は動的なポートを使用するアプリケーションには対応できず，実際のトラヒックの 30%以下しか識別できない．もう一つの手法であるトラヒック解析による識別は, ほぼ 100%の識別の精度で識別できるが, 解析負荷が高いこと，暗号化トラヒックの解析が難しいという課題がある．

これら既存手法の課題を解決する方法として, 近年トラヒックフローの統計情報を利用した識別手法が提案されている．提案されているのは，主に機械学習を利用した手法であり，DNN を使って Voip のようなストリーミングアプリケーションの識別を行なった研究が無いことを示した．

第 3 章では，本研究で利用したデータセットと，DNN による具体的な識別手法について述べた．Deep Neural Network, Dropout, Cross Validation, Hyper Parameter 探索について詳しく説明した．

第 4 章では，本研究で利用した実験機材と環境について述べ，第 3 章で説明した手法を用いた結果を示し，高い識別性能と汎化性能を実現できていることを確認した．同時に Neural Network の可視化を行い，各トラヒックフローがクラスタリングされて

いること，どの統計情報が識別に有効か示すことができた．また同じデータセットを用いている既存の機械学習の手法を用いた論文と比較し，同等以上の精度であることを示した．ポート番号，ペイロード解析手法も含めた比較を表 5.1 に示す．

表 5.1：手法の比較

手法	精度	計算量	暗号化・ストリーミング	プライバシー
ポート番号	30%以下	少	非対応	一部侵害
ペイロード解析	ほぼ 100%	多	非対応	侵害
C4.5	70%～99%	少	対応	問題無し
提案手法(DNN)	99%以上	やや多い	対応	問題無し

第 4 章の後半では，実時間での識別の処理性能について考察し，処理性能向上のために，DNN モデルの効率化が必要なことを考察した．

参考文献

[1] Traffic classification, https://en.wikipedia.org/wiki/Traffic_classification

[2] IANA (IANA), port number assignments. http://www.iana.org/assignments/port-numbers.

[3] T.Karagiannis, K.Papagiannaki, and Michalis Faloutsos. "Blinc: Mul- tilevel traffic classification in the dark." SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications. New York, NY, USA: ACM Press, pp. 229–240, 2005.

[4] Velan, Petr, et al. "A survey of methods for encrypted traffic classification and analysis." International Journal of Network Management 25.5 (2015): 355-374.

[5] Zhou, Dingding, et al. "Research on Traffic Identification Based on Multi Layer Perceptron." TELKOMNIKA (Telecommunication Computing Electronics and Control) 12.1 (2014): 201-208.

[6] Alshammari, R.; Zincir-Heywood, A.N., "Can Encrypted Traffic be identified without Port Numbers, IP Addresses and Payload Inspection?", Journal of Computer Networks, Elsevier, 2011.

[7] Alshammari, Riyad; Zincir-Heywood, A. Nur, "Investigating Two Different Approaches for Encrypted Traffic Classification", PST '08. Sixth Annual Conference on Privacy, Security and Trust, 2008, vol., no., pp.156-166, 1-3 Oct. 2008.

[8] Alshammari, R.; Zincir-Heywood, A.N., "A flow based approach for SSH traffic detection," Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on , vol., no., pp. 296-301, 7-10 Oct. 2007.

[9] Alshammari, Riyad; Zincir-Heywood, A. Nur; , "An Investigation on the Identification of VoIP traffic: Case Study on Gtalk and Skype," 6th International Conference on Network and Services Management CNSM 2010), Niagara Falls, Canada, October 25-29, 2010.

[10] ZFone, https://en.wikipedia.org/wiki/Zfone

[11] Primus https://www.andone.co.jp/ip-pbx-primus/sdk/

[12] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." Journal of Machine Learning Research 15.1 (2014): 1929-1958.

[13] Kohavi, Ron. "A study of cross-validation and bootstrap for accuracy estimation and model selection." Ijcai. Vol. 14. No. 2. 1995.

[14] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13:281–305, 2012.

[15] Keras, https://keras.io/ja/

[16] TensorFlow, https://www.tensorflow.org/

[17] Skype, http://www.skype.com/useskype/.
[18] Google talk (gtalk), http://www.google.com/talk/.
[19] AUC, https://en.wikipedia.org/wiki/Receiver_operating_characteristic
[20] Wireshark, https://www.wireshark.org
[21] Kenjiro Cho(IIJ), Kensuke Fukuda(NII/PRESTO JST), Hiroshi Esaki(東京大学), Akira Kato(慶応義塾大学), Observing Slow Crustal Movement in Residential User Traffic, ACM CoNEXT2008, Dec 10-12, 2008, Madrid, SPAIN
[22] Cisco800 シリーズ ルータ, http://www.cisco.com/c/ja_jp/products/routers/800-series-routers/index.html