

深層学習を用いた無線LANパケット解析に基づく 輻輳の予測

山本 葵¹ 山口 実靖² 神山 剛³ 小口 正人¹

概要：近年、世界中に増え続けているスマートフォン、タブレット端末は機能や性能も強化されている。気軽にネットワークにアクセスし、動画やゲームなどのデータ通信を楽しむことが出来るようになり、大容量かつ高速な通信に対する需要は増大している。しかし有線接続に比べ低帯域かつノイズの多い無線接続においては、膨大なパケットが通信中に無線 LAN アクセスポイントに蓄積され、その結果輻輳が発生してしまうという問題も生じている。本研究では輻輳発生前に制御を加え無線 LAN AP の輻輳を回避することを最終目的とし、本稿では目的達成のため輻輳の予測を行う。Android 端末を用いて無線 LAN 通信を行い、アクセスポイント周りのパケットをキャプチャした。そのパケットを深層学習の LSTM モデルを用いて解析し無線 LAN 通信時のトラフィックの予測性能を評価した。

Congestion Prediction Based on Analysis of Packet of Wireless Communication Using Deep Learning

AOI YAMAMOTO¹ SANEYASU YAMAGUCHI² TAKESHI KAMIYAMA³ MASATO OGUCHI¹

1. はじめに

モバイルネットワークにアクセスするスマートフォンやタブレット端末などのワイヤレスデバイスは世界中で増加し続けている。毎年多様な形状のワイヤレスデバイスが市場に登場し、これらの機能や性能も強化され進化し続けている。全世界のワイヤレスデバイス数は、2020 年までには 116 億にまで増加し、1 人あたりのデバイス数は 1.5 台になるという予想もなされている [1]。

端末自体の高機能化、高性能化は気軽にホームページの閲覧や音楽、動画やゲームなどのデータ通信を行うことを容易にしている。このようなデータ通信は大容量になることも多くあり得る。大容量かつ高速な通信に対する需要は増大すると考えられ、全世界のモバイルトラフィック量は 2015 年から 2020 年の間に約 8 倍も増加すると予想されている [1]。

このことから大量のデータ通信がワイヤレスデバイスから発生しており、電波帯域は圧迫され、足りなくなりつつ

ある。この問題を解決しようと携帯電話事業者は基地局の設置とともにデータオフロード、その中でも Wi-Fi オフロードを推進している。混雑しているワイヤレスデバイス-基地局間の無線部分のデータ通信を、Wi-Fi 経由にすることでデータを有線接続の固定回線などに誘導し、無線部分の通信量を減らそうとしている。実際に街中の施設では Wi-Fi オフロードを推進している携帯電話事業者などが配布した AP をみる機会も多くあるだろう。

このような背景から今後無線 LAN へアクセスする端末数やトラフィック量が増加し、ワイヤレスデバイス-アクセスポイント (AP) 間においても大量のデータ通信が発生し帯域が圧迫されることが考えられる。ワイヤレスデバイスから送信されるパケットを処理する AP の負荷が増大し、輻輳が起こる頻度も多くなると考えられる。

TCP プロトコルは、様々な目的でのデータ通信において標準的に利用されているプロトコルの 1 つであり、ネットワークの公平かつ効率的な利用のため、TCP には輻輳制御アルゴリズムが備わっている。Linux や Android が採用しているロスペースアルゴリズムはパケットのロスを観測し、ロスが増加すると輻輳が発生したとして送信量を抑

¹ お茶の水女子大学

² 工学院大学

³ 九州大学

えるというアルゴリズムである。しかし、より高いスループットを得るためにアグレッシブにパケットを送信するため、有線接続に比べて低帯域かつノイズの多い無線接続環境においてはその手法によって膨大なパケットが蓄積されてしまうという問題が生じることがある。この問題の解決法としては、高速通信の規格化があり、使用可能な周波数帯の増加や伝送速度の向上があれば輻輳は起きにくくなるしかし規格が広く普及するには時間がかかり、実際街中では狭い帯域を取り合っているのが現状である。

よって本研究では無線 LAN AP の輻輳を回避することを最終目的とし、本稿ではこの目的を達成するために輻輳を予測できるか検証する。輻輳の発生が事前に検知できれば、CWND の補正をなどによって、輻輳発生前にパケットの送信量を抑えることができる。その結果パケットロスを発生させることなく、また無線 LAN AP に接続している全端末が平等に安定したスループットを確保できる輻輳制御が実現できる可能性がある。

本稿では、無線通信のトラフィックを解析し、輻輳の極めて早期における検出、予兆の発見をし輻輳の予測を行う。具体的には、Android 端末を用いてデータ通信を行い、無線 LAN AP 周りのパケットをキャプチャデバイスを用いて取得し、入力セットとする。また通信中、AP 宛に ping コマンドを用いて端末-AP 間の RTT 値を測定、記録する。これらのデータを用いて深層学習を行い、トラフィックの予測が可能であるか検証した。

2. 関連研究

2.1 カーネルモニタ

カーネル内部の処理は通常バックグラウンドで進められているため、通常ユーザ空間からその処理の様子を監視することはできない。そこで先行研究 [2] によりカーネル内部の情報をるためにカーネルモニタというツールが開発された。TCP のソースコードにモニタ関数を挿入し、カーネルを再構築することで、メモリからログを得ることができる。これにより輻輳ウインドウサイズや往復遅延時間 (RTT)、各種エラーイベントの発生タイミングなどの TCP パラメータをリアルタイムにモニタすることができる。

2.2 輻輳制御ミドルウェア

先行研究 [3][4] で開発された輻輳制御ミドルウェアは、カーネルモニタをベースとしたシステムであり、Android 端末間の連携した制御を目的としている。Android 端末間でこれから送信するセグメント数を表す輻輳ウインドウ値を通知し、周辺端末の通信状況を把握する。さらに周辺端末から受けた情報に基づき、輻輳ウインドウの上限値を自動で算出し補正することで、端末間で可用帯域を公平に分け合う。これにより無線 LAN AP に於ける ACK パケットの蓄積を回避する。

さらに [5] では輻輳制御ミドルウェアの改良を行っている。システムの発動タイミングを調整し多くの端末が同時に通信するときの全体の通信速度と公平性の向上を可能にした。

3. 深層学習

深層学習はニューラルネットワークの階層を深めたアルゴリズムで、機械学習を実装するための 1 つの手法である。機械が自分自身で特徴量を抽出できるようになり、また階層を深めることで精度が大幅に向上了。これにより現在は第 3 次 AI ブームとも言われている。代表的な実用例は、郵便局で郵便番号を認識して選別する際に使われる文字認識、Amazon の売り上げを大きくあげたことで有名な商品レコメンドシステム、自動車の運転支援システムなど幅広く使われている。深層学習で用いられるモデルをさらに詳しく説明する。

3.1 ニューラルネットワーク (NN)

人間の脳内にある神経細胞とその繋がり、神経回路網を人工ニューロンという数式的なモデルで表現したもので、一般的なものは図 1 のように多数の層から一方향へ情報が伝搬されるようなモデルがよく使われる。層と層の間にはニューロン同士の繋がりの強さを表す重みがある。NN の発案はコンピュータが普及し始めた時と言われており、当時のコンピュータは非力で膨大な計算を行えるほどの容量や計算能力ではなかったが、近年のコンピュータのスペックの向上により深層学習が容易となった。

3.2 リカレントニューラルネットワーク (RNN)

RNN(図 2) は、入力データは互いに独立であると仮定されていた NN と違い、時系列の流れに意味を持つデータの予測や分類に用いられるモデルである。RNN は以前に計算された情報を覚えておくための記憶力を持っている。理論的には長いデータを記憶し、利用することが可能だが、実際は 2, 3 ステップ前くらいの記憶しか維持できないという欠点がある。

3.3 ロングショートタームメモリネットワーク (LSTM)

LSTM(図 3) は RNN の拡張として登場した、時系列データに対するモデルである。LSTM は RNN の隠れ層のユニットを LSTM block と呼ばれるメモリと 3 つのゲートをもつブロックに置き換えることで実現された。その最も大きな特徴は RNN ではできなかった長期依存が可能であるということである。

本研究は時系列データであるパケットの解析であるため、この LSTM をモデルとした深層学習を行う。

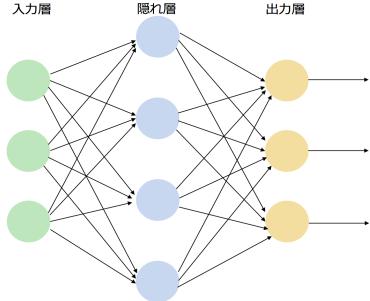


図 1 NN

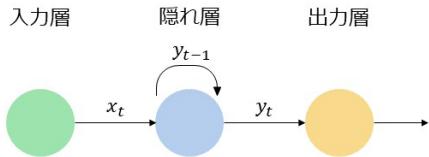


図 2 RNN

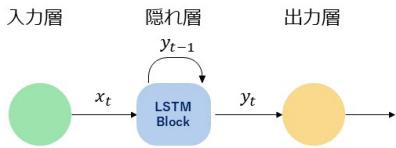


図 3 LSTM

4. 正解データの検討

本章では各パラメータの振る舞いを観察することにより深層学習をする際の正解データをどのように設定するか検討を行う。輻輳を示す値としてどのパラメータを採用すべきか実験を行い決定する。

各パラメータを観察するために、先行研究 [2] にて開発されたカーネルモニタを導入した Android 端末 5 台を用いて通信を行い、通信時の TCP パラメータを取得する。またその間、PC から ping コマンドを用いて PC と AP 間の RTT 値を測定する。実験環境は図 4 である。輻輳の発生を観察するために 60 秒間パラメータを取得した。0-20 秒は Android 端末からは一切パケットを送信はせず、20 秒付近で 5 台の Android 端末から一斉に iperf によりサーバにパケットを送信した。全端末においてパケットを送信する、しないの極端なデータ通信を行い輻輳を発生させた。この実験により 3 つのパラメータを取得し、グラフとして出力したものが図 5-図 7 である。

4.1 Android 端末の CWND 値

図 5 は通信を行った Android 端末 5 台の輻輳ウインドウ (CWND) の振る舞いを測定した結果である。20 秒まで

は通信を行っていないので値は 0 で一定だが、パケットを送信し始めた 20 秒付近からは各端末が自由にパケットを送出しており、TCP が各端末を制御している様子が見て取れる。Android 端末からパケットを送信し始めた 20 付近以降、輻輳を検知した TCP が CWND の値を変えて制御を行っている。Android が採用している輻輳制御アルゴリズムであるロスベースアルゴリズムは、パケットロスの検知により輻輳と判断している。よって図 5 の CWND の振る舞いをみると、各端末が自由に大量のパケットを出し、パケットロスが起こるまで指数関数的に CWND が増加している。その結果輻輳が発生し、パケットロスを検知した TCP が CWND の値を変更することによる制御が行われていることが確認できる。

4.2 Android 端末-Server 間の RTT 値

図 6 は Android 端末-Server 間の RTT 値の振る舞いを測定した結果である。パケットを送信し始めた 20 秒付近から徐々に RTT 値が増加している。パケットを送信し始めた 20 秒付近ではまだキューが溜まっていないため適切にパケットを処理できるため RTT 値は大幅には増加していないが、各端末は 20 秒以降は大量にパケットを送出し続けているため、40 秒以降でパケットの処理が追いつかず RTT 値が大幅に上昇していると考えられる。

4.3 PC-AP 間の RTT 値

図 7 は PC-AP 間の RTT 値の振る舞いを測定した結果である。折れ線がグラフから消えている部分は RTT 値を測定するためのリクエストに対して応答がなくタイムアウトしたものである。応答が返って来ないということは AP が非常に混みあっており輻輳が発生していると考えられる。

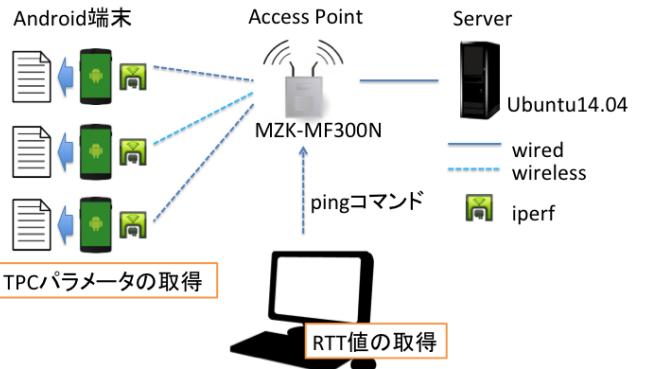


図 4 基礎実験環境

4.4 正解データの決定

実験により正解データとしては、Android 端末でデータ通信中に PC から ping コマンドを用いて測定した PC と AP 間の RTT 値を用いる。図 5 と図 7 より輻輳が発生している時に PC-AP 間の RTT 値も上昇しており、RTT 値の上昇は AP の混雑具合や輻輳の発生を表していると考えられるからである。

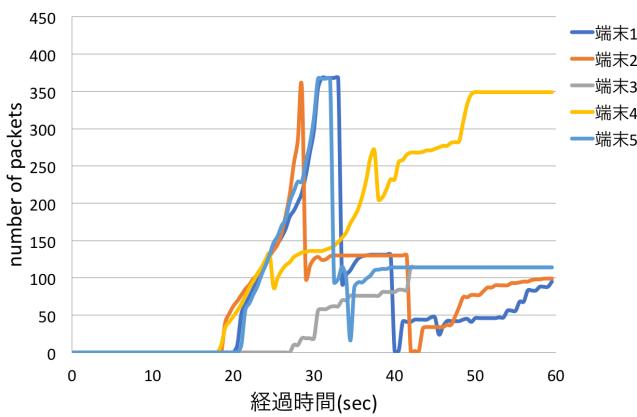


図 5 5 台同時通信における Android 端末の CWND の推移

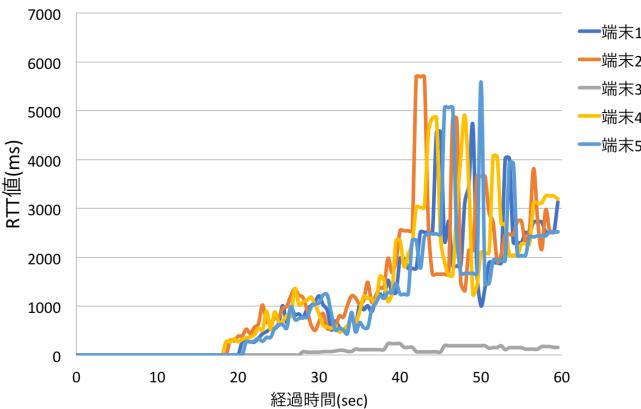


図 6 Android 端末-Server 間の RTT 値の推移

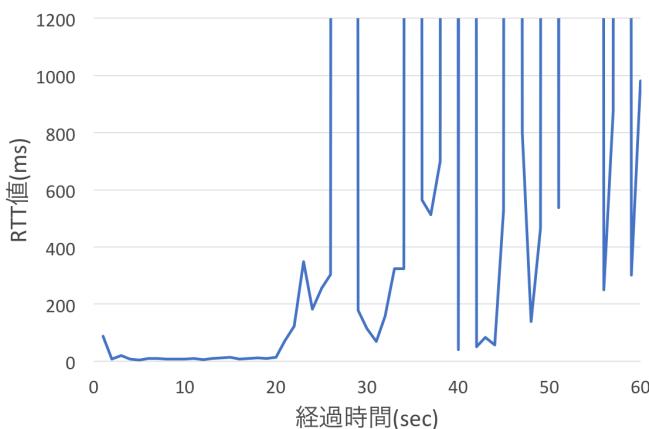


図 7 PC-AP 間の RTT 値の推移

5. 実験

本章では様々な状況下で Android 端末によるデータ通信を行なった時に輻輳が予測できるか評価実験を行なった。入力データの情報量や学習時間などを変更してどの程度予測できるか実験を行う。

5.1 データセット

深層学習に用いるデータセットについて説明する。正解データには 4 章より PC-AP 間の RTT 値を使用する。

5.1.1 入力データ

入力データとして無線 LAN AP 周辺のパケットを使用する。用いるキャプチャデバイスは米国、riverbed 社の AirPcap[6] である。AirPcap は wireshark 統合型のワイヤレストラフィックパケットキャプチャデバイスで、制御、管理、データの各フレームを含む、IEEE802.11 a/b/g/n のトラフィックをキャプチャできるデバイスである。

5.2 深層学習用フレームワーク

今回の実験に用いた深層学習用フレームワークは Chaier である。これは Preferred Networks 社が開発し、2015 年に公開された Python のライブラリである。特徴として”Flexible”，”Intuitive”，”Powerful”を掲げている [7]。

5.3 実験環境

5.1 章のデータを取得し実験を行う。実験環境を図 8 に示した。1 台の AP に接続した複数台の Android 端末から iperf[8] によって、データをサーバに送信する。それと同時に PC で上記のデータを取得する。その後、適切なデータ形式に加工し、LSTM モデルを用いた深層学習をおこなう。

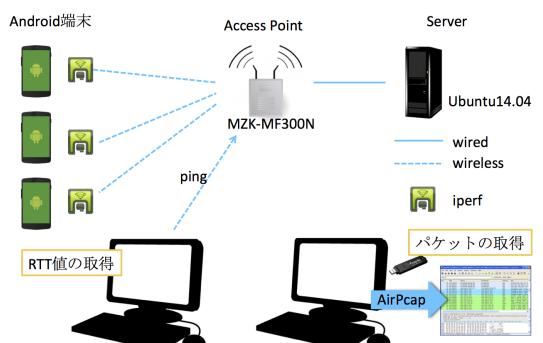


図 8 実験環境

5.4 実験 1

初めに 2 つの実験を行なった。スマートフォン 2 台、タブレット 2 台の計 4 台の Android 端末を用いて通信を行い、87 秒間のデータを用いてデータセットとした。2 つの実験は入力データに違いがあり、その他の条件は同じであ

表 1 実験機器の性能

Android	Model number	Nexus S	Nexus 7(2013)
	Firmware version	4.1.1	6.0.0
	CPU	1.0 GHz Cortex-A8	Quad-core 1.5 GHz Krait
	Memory(Internal)	16 GB, 512 MB RAM	16 GB, 2 GB RAM
	WLAN	Wi-Fi 802.11 b/g/n	Wi-Fi 802.11 a/b/g/n
server	OS	Ubuntu 14.04 (64bit) / Linux 3.13.0	
	CPU	Intel(R) Core(TM)2 Quad CPU Q8400	
AP	Main Memory	8.1GiB	
	Model	MZK-MF300N(Planex)	
	Support Format	IEEE 802.11 n/g/b	
	Channel	13	
Frequency Band	2.4 GHz(2, 1412-2, 472 MHz)		

表 2 解析に用いた PC の性能

PC	OS	14.04.1-Ubuntu
	CPU	Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz
	GPU	GeForce GTX 1080
	Memory	2.4 GHz(32GB)

る。学習回数を表す epoch 数は 2000 に設定し、どちらの結果のグラフも t から t+9 秒の入力データを用いて t+10 秒の正解データである RTT 値を予測したものである。実験結果を図 9, 10 に示す。オレンジの線が正解の RTT 値、青の線が予測した値である。

5.4.1 結果 1-1(一次元入力データ)

1 つ目の実験は第 1 段階として、入力データを表 3 のように 1 秒間の平均データ量のみの一次元データを使用して深層学習を行なった。結果は図 9 のように全体的に振れ幅が少なく、平坦なグラフが output された。70 秒付近の 1 番急激に RTT 値が増加している部分は少し予測もついているが、これは十分に予測ができるといえず、この程度の予測では AP が混み合っているかの判断材料にすることはできないと考えられる。

表 3 実験 1 データセット

入力データ	正解データ
1 秒間の平均データ量	RTT 値

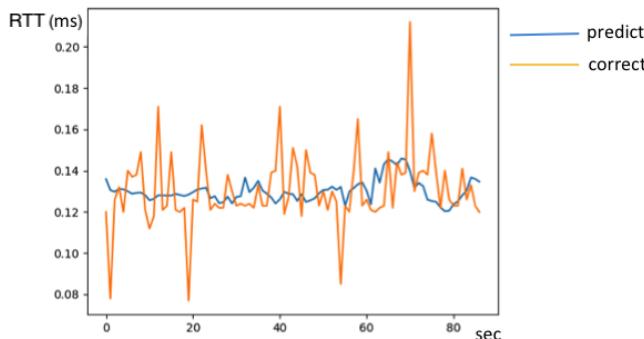


図 9 実験 1 結果 (一次元入力データ)

5.4.2 結果 1-2(五次元入力データ)

実験結果 1-1 から、さらに精度の向上を目指すため、入力データを先ほどの一次元から五次元に増やし再度実験を行なった。データセットの詳細は表 4 である。結果は先ほどよりも良いものとなった。特に 10 秒付近の RTT 値が急

増している箇所や、65 秒付近ではうまく予測が行われていることがわかる。さらに実験結果が正解とどれくらい離れているかを表す loss 値をみていく(図 11)。これは 0 に近くほど正解に近いということであるが、loss の値は順調にさがっており学習がうまくいっていることがわかる。

表 4 実験 2 データセット

1 秒間	入力データ		正解データ
	時間	パケット数	平均データ量
	送信機器の台数	受信機器の台数	RTT 値

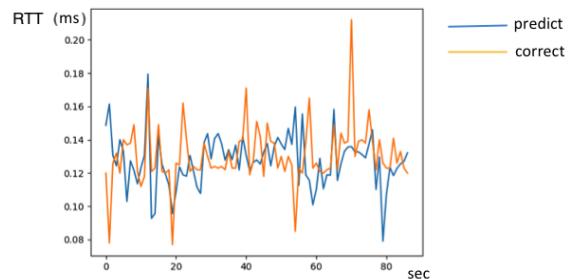


図 10 実験 2 結果 (五次元入力データ)

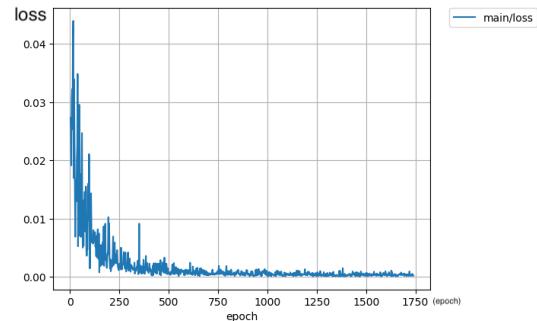


図 11 実験 2 loss

5.5 実験 2

実験結果 1-2 より、五次元入力データによる学習は予測ができる可能性があることがわかった。そこで精度の向上を目指すため、長時間のデータによる学習を行なった。さらにこの学習の汎化能力を検証するためテストデータによる検証も行った。この実験にはスマートフォン 7 台、タブレット端末 2 台の計 9 台の Android 端末を用いた。データセットは全て新規に作成し、学習用データセットとして 609 秒間のパケット情報と RTT 値を、テスト用データセットとして 353 秒のパケット情報と RTT 値を使用した。入力データの詳細は先ほどと同じく表 4 である。全ての結果のグラフも同じく t から t+9 秒の入力データを用いて t+10 秒の正解データである RTT 値を予測したものであり、オレンジの線が正解の RTT 値、青の線が予測した値である。

5.5.1 結果 2

結果は、1枚のグラフで出力するとデータ数が多く見づらいため、学習データによる予測結果は3つに、テストデータによる予測結果は2つに分割した。学習データによる予測結果は、図12の0-200秒の範囲では合っている部分もあるが、図13のRTT値が安定して低い300-375秒付近は全く予測できていなかった。テストデータによる予測結果は、図15の0-25秒、図16の175-225秒付近はうまく予測が行われているが、RTT値が急増していたり、反対に安定して低い部分は予測できていなかった。

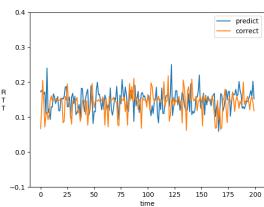


図 12 学習データによる
予測結果 1

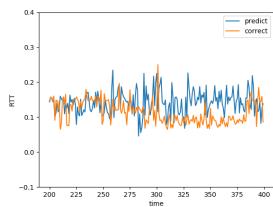


図 13 学習データによる
予測結果 2

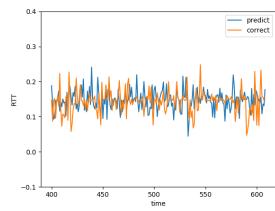


図 14 学習データによる予測結果 3

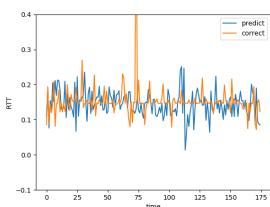


図 15 テストデータによる
予測結果 1

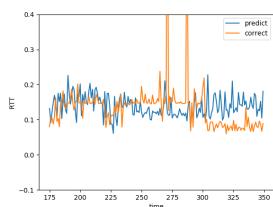


図 16 テストデータによる
予測結果 2

5.6 実験 3

実験1、実験2より長時間のデータを学習させるより入力データの情報を多くした方が良い結果を得られることがわかった。実験3ではよりシンプルな実験を行い予測精度の向上を目指す。この実験ではスマートフォン4台、タブレット端末1台の計5台のAndroid端末を用いた。

学習データとして130秒のデータを作成した。130秒のうち、0-29秒はAndroid端末からは一切パケットを送信せず30秒で一斉にAndroid端末からiperfを用いてデー

タ通信を開始し、70秒間パケットを送信し続けた。その後30秒は再度一切パケット送信をしない時間を設けた。

この実験においても学習の汎化能力を検証するためテストデータによる検証も行った。テストデータは新規に作成した120秒のデータである。今回は20-80秒の60秒間のみパケットを送信しており他の時間は一切パケットを送信していない。

データセットとしては実験2の入力データに加えカーネルモニタから得られるTCPパラメータを追加した。Android端末5台から得られる各TCPパラメータのうち、Android端末-Server間のRTT値とCWNDの平均値を追加した。詳細は表5である。正解データとなるPC-AP間のRTT値は測定中、リクエストがタイムアウトとなってしまうことが複数回あった。pingコマンドによるRTT値の測定はリクエストに対する応答が1000ms以上ない場合タイムアウトとなる。そこでタイムアウトした場合のRTT値を1500msとし正解データとした。結果のグラフは先ほどと同じくtからt+9秒の入力データを用いてt+10秒の正解データであるRTT値を予測したものであり、オレンジの線が正解のRTT値、青の線が予測した値である。

表 5 実験3データセット

	入力データ	正解データ
1秒間	時間 パケット数 平均データ量 送信機器の台数 受信機器の台数 平均 RTT 値 (Android 端末-Server 間) 平均 CWND	RTT 値 (PC-AP 間)

5.6.1 結果 3

学習データによる予想結果は図17となった。正確な値は予測できなかったが、RTT値が上昇するタイミング、下降するタイミングは予測できている。

テストデータによる予測結果は図18である。やはり正確な値の予測は出来ていないが、テストデータによる実験でもRTT値の上昇、下降タイミングは数秒の誤差で予測できている。実験2のテストデータによる予測結果よりも良い結果となった。学習データとはパケットを送信するタイミングやデータセットの時間も異なっているため、汎化能力もあると考えられる。

6. まとめと今後の課題

本研究では、APに接続する端末が複数台通信を行い、輻輳がおこるであろう場合において、輻輳の極めて早期の検出、予兆の発見を行うために、深層学習を用いてトラフィックの予測を行なった。具体的には無線LANAPに接続したAndroid端末を用いてデータ通信を行い、そのパケットをデバイスを用いてキャプチャする。そのパケット情報を入力データとし、またAPの混雑を表すであろう

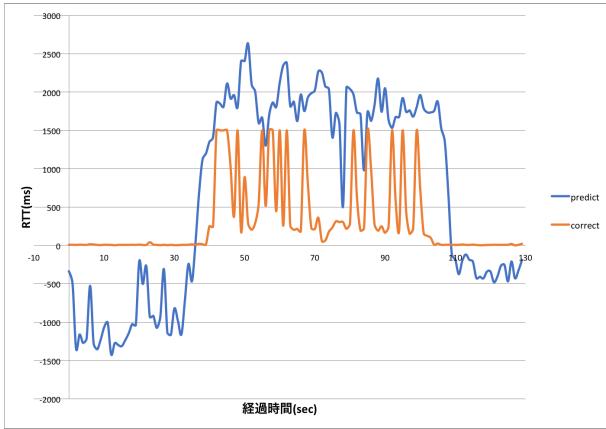


図 17 実験 3 学習データによる予測結果

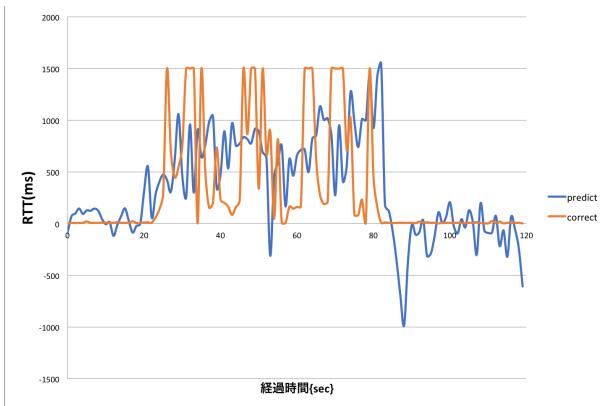


図 18 実験 3 テストデータによる予測結果

AP 宛の RTT 値を正解データに用いて深層学習を行なった。深層学習はフレームワークに Preferred Networks 社の Chainer を使い、時系列データに適する LSTM モデルを使用してトラフィックの予測が行えるか実験を行なった。その結果から入力データの情報量は非常に重要で、多ければより正確な予測を行えることがわかった。

しかし、五次元入力データは一次元入力データとの比較では精度が向上していると言えるが、部分的な範囲に留まっており、全体的にみるとまだ十分な精度とは言えない。この学習用データセット、テスト用データセットを作成するにあたっては、複数の Android 端末を用いて、全台数が通信をしている状態や反対にほとんど通信している端末がない状態などランダムに状態を変化させ様々な状況でのパケット情報と RTT 値の増減が現れるように作成したが、まだあらゆる状況下でのデータが十分ではない可能性があり、10 分程度のパケット情報では精度には限界があることがわかった。

また、Android 端末からパケットを一切送信しない時と一斉に送信をする場合で実験を行った。正確な値まで予測はできなかったが、PC-AP 間の RTT 値が上昇するタイミング、下降するタイミングは予測することが出来ていた。この実験においてはテストデータによる検証においても良

い結果となったため、極端なデータ通信を行い、一斉に大量のパケットを送信することで輻輳を発生させるというよりシンプルな状況においては予測が可能であると考えられる。

今後の課題としては様々な状況下でデータ通信を行なっている時の予測精度の向上が 1 番に挙げられる。方法として 1 つ目は入力データの情報量を増やすことである。今回の実験で入力データを一次元から五次元にしただけで精度の向上がみられた。よってさらに入力データの情報量を増やせばさらに精度の向上が見込まれる。キャプチャデバイスから得られる情報は非常にたくさんあり、多数のレイヤの情報がとれる。今回はその中から一部の情報を集計して用いたが、得られる情報全て、各パケット 1 づつを入力データにしてトラフィックを多面的に捉え学習、予測をしていきたい。また正解データも工夫の余地があるのでないかと考えられる。また、長時間の学習としては 609 秒間という時間のデータで実験を行なったが、数十分、数時間、数日とさらに長時間のデータを集めてあらゆる状況においてのデータを用いて十分に学習をさせていきたい。端末が一斉に通信を行なった場合だけでなく、ランダムに通信を行なっている場合においても精度よく予測ができるように学習を行って行きたい。

また今後輻輳制御を行っていくことを見据え、計算時間も考慮に入れて実験を行っていきたい。いくら予測精度が良くても、予測するために計算している間に結果の予測時刻を過ぎてしまっては意味がない。どの程度先の時刻まで精度よく予測できるかという実験も今後行っていく予定である。

謝辞

本研究は一部、JST CREST JPMJCR1503 の支援を受けたものである。

参考文献

- [1] Cisco Visual Networking Index, https://www.cisco.com/c/ja/jp/solutions/collateral/service-provider/visual-networking-index-vni/white-paper_c11-520862.html
- [2] Kaori Miki, Saneyasu Yamaguchi, and Masato Oguchi: "Kernel Monitor of Transport Layer Developed for Android Working on Mobile Phone Terminals," Proc. ICN2011, pp.297-302, January 2011.
- [3] Hiromi Hirai, Saneyasu Yamaguchi, and Masato Oguchi: "A Proposal on Cooperative Transmission Control Middleware on a Smartphone in a WLAN Environment," Proc. IEEE WiMob2013, pp.710-717, October 2013.
- [4] Ai Hayakawa, Saneyasu Yamaguchi, Masato Oguchi: "Reducing the TCP ACK Packet Backlog at the WLAN Access Point," Proc. ACM IMCOM2015, 5-4, January 2015.
- [5] Ayumi Shimada, Saneyasu Yamaguchi, and Masato

- Oguchi: "Performance Improvement of TCP Communication based on Cooperative Congestion Control in Android Terminals," Proc. ACM IMCOM2018, 3-2, January 2018.
- [6] riverbed, <https://www.riverbed.com>
 - [7] Chainer, Framework for Neural Networks, <https://chainer.org/>
 - [8] Iperf For Android Project in Distributed Systems, <http://www.cs.technion.ac.il/~sakogan/DSL/2011/projects/iperf/index.html>