機械学習を用いたAndroid端末上での時系列データ予測に向けて

佐藤 里香[1] 山口 実靖[2] 神山 剛[3] 小口 正人[1]

概要：近年スマートフォンの普及が急速に進み, 大容量のデータ通信が行われるようになった. それに伴って無線 LAN への接続需要が高まってきているが, 無線環境下でのトラフィックの輻輳やパケットロスが問題となっている. 突発的に生じる輻輳は一度起こると制御が難しい上, 制御しようとしてさらに輻輳が悪化してしまうことがあるため, 輻輳が起こる前にそれを予測していくことが望ましい. そこで本研究では, スマートフォン端末上で事前にトラフィックの輻輳を予測し輻輳を制御, 回避することを最終目的とし, 本稿ではスマートフォン用深層学習実装 TensorFlow Lite を用いて端末内で学習モデルを用いた予知を行う手法に注目, その実現可能性について考察する.

Toward Time Series Data Prediction on Android Devices by Using Machine Learning

RIKA SATO[1] SANEYASU YAMAGUCHI[2] TAKESHI KAMIYAMA[3] MASATO OGUCHI[1]

1. はじめに

近年, スマートフォンの高性能化, 多機能化により新たな機種が続々と登場し, 先進国のみならず世界中でスマートフォンの出荷台数が急激に増加している. ここ数年は少し落ち着いてきたものの, 次世代通信方式 5G 導入に伴い, 今後出荷台数がさらに増えるとの予想もされている [1].

従来のフィーチャーフォンでは通話やメールが使用目的の大半であったが, スマートフォンでは Web や動画の閲覧, SNS をはじめとしたアプリケーションの利用などその用途が多岐に渡っており, 大容量のデータ通信が行われるようになった.

このような事情から無線 LAN へのアクセスが増加している. 最近では飲食店や公共交通機関でのフリー Wi-Fi スポットといった公衆無線 LAN サービスが増えてきていたり, 家庭向けの無線 LAN 機器の普及も進んでいる.

しかし, 無線 LAN は帯域の狭さやノイズの影響など障害が多く, 有線接続に比べると脆弱であるため無線環境下でのトラフィックの輻輳やパケットロスといった問題が生じている. この輻輳は突発的に生じ, 一度起こると制御が難しい上にコントロールしようとしてさらに輻輳が悪化してしまうことがあるため, 輻輳が起こる前にそれを予測していくことが望ましいと考えられる.

また, 輻輳の予知に関して, データを端末外に出すセキュリティ上の問題やデータ転送に要する時間等の課題から, 端末内での処理が好ましいと言える.

そこで本研究では, Android 端末上でトラフィックの輻輳を事前予測, 制御を行って端末からの送信データ量を調整することにより輻輳を回避することを最終目標とする. そのためにはスマートフォンのようなリソースの限られた端末で, 十分な精度の予測をリアルタイムに行うことが必要となる. そこでまずサーバ機などの性能の高いマシン上でトラフィックの輻輳を深層学習により予測し, そのモデルを Android 端末に導入してサーバ機と同等の精度や処理速度で予測できるようにすることを目指す.

本稿ではまず端末におけるトラフィックをサーバ機上で深層学習により予測した結果を示し, またその学習モデルをスマートフォン端末に組み込める形式に変換できること

[1] お茶の水女子大学
Ochanomizu University
[2] 工学院大学
Kogakuin University
[3] 九州大学
Kyushu University

を確認する.

2. 関連研究

2.1 カーネルモニタ

先行研究 [2] においてカーネルモニタと呼ばれるツールが開発された. このツールは, 通信時にカーネルのコードのどの部分がいつ実行され, その結果カーネル内部のパラメータの値が どのように変化したかを記録することができるというものである. このカーネルモニタにより正常動作時のカーネルの振る舞いを知ることができ, さらに通信において生じた問題を特定し原因を調べることも可能である. このツールを Android に組み込むことでリアルタイムに解析を行うことができる.

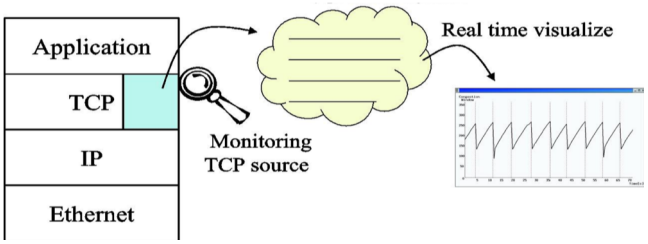

図 1 カーネルモニタ

2.2 輻輳制御ミドルウェア

2.1 のカーネルモニタをもとに, 先行研究 [3] では輻輳制御ミドルウェアというツールが開発され, 先行研究 [4] ではその改良が行われた. このツールは複数台の Android 端末が同一の無線 LAN アクセスポイントに接続する際に, カーネルモニタによって取得した情報により端末間で輻輳の状態を把握し, 協調して輻輳を制御するということを目的としている.

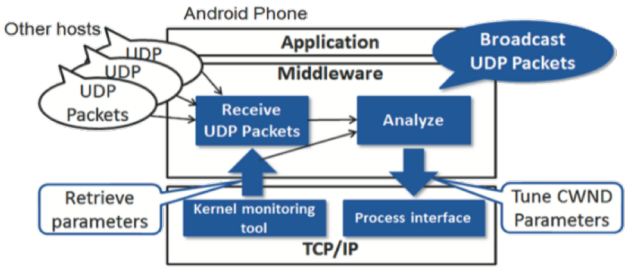

図 2 輻輳制御ミドルウェア

これらの先行研究は, 本研究と同様 Android 端末上で無線 LAN トラフィックの輻輳を制御することを目的としたものであるが, 輻輳を認識してから制御を行う先行研究に対し, 本研究では深層学習により輻輳が生じる前に輻輳を予知し, 制御を行うという点で先行研究とは異なる.

3. Android

Android[5] は Google 社が開発したモバイルオペレーティングシステムであり, 現在世界トップのシェアを誇っている [6]. Android の大きな特徴としてはオープンソースであるということや, Windows や Linux, MacOS といった様々な環境でアプリケーションの開発を行うことができるといったことが挙げられる. このような特性から本研究ではスマートフォンの OS として Android を対象に研究を行う.

4. ディープラーニング (深層学習)

ディープラーニング (図 3) は, ニューラルネットワークにおいて隠れ層の数を増やし階層を深くしたディープニューラルネットワークを学習する手法である. コンピュータが自らでデータ の特徴を捉え, より効率的で高い精度での学習が可能となり, 自動運転や音声入力など様々な分野での実用化が進んでいる.

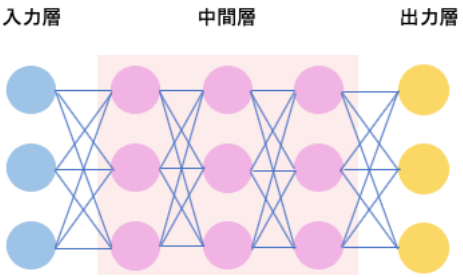

図 3 ディープラーニングのモデル

4.1 RNN(recurrent neural networks)

ディープラーニングの一種である RNN(リカレントニューラルネットワーク, 図 4) は, データ間に繋がりのある時系列データの扱いに特化したモデルであり, 前の時刻の中間層を次の時刻の入力と合わせて学習に用いることで, 時系列情報を考慮した解析を行うことができる. 音声認識, 動画認識や自然言語処理に長けたアルゴリズムであるが, 長期の依存関係にあるデータの処理ができないという問題点もある.

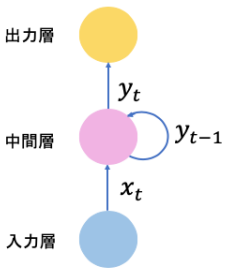

図 4 RNN のモデル

4.2 LSTM(Long short-term menory)

LSTM(Long short-term memory, 図 5) は RNN を拡張したもので, ニューロンの一つ一つを LSTM ブロックと呼ばれる仕組みに置き換えることで長期の時間依存性も短期の時間依存性も学習することができるアルゴリズムとなっている.

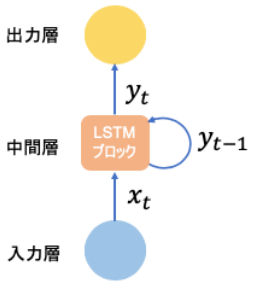

図 5 LSTM のモデル

5. TensorFlow Lite

TensorFlow Lite[7] は Google の機械学習向けソフトウェアライブラリである TensorFlow のモバイル環境向けのライブラリである. TensorFlow Lite では, TensorFlow によりトレーニングされたモデルを TFLite Converter を使って TensorFlow Lite 形式に変換し, デバイスに組み込むことでデバイス上で推論を行うことができる.

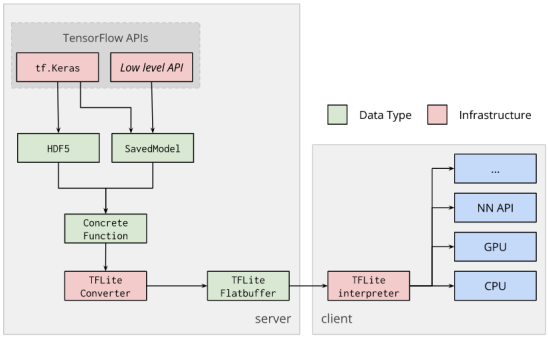

図 6 学習モデル変換の流れ

5.1 TFLite Converter

TFLite Converter は TensorFlow の機械学習モデルを入力として TFLite FlatBuffer File を生成するコンバータ (変換器) である. 図 6 の左側にあるように, サーバ側で TensorFlow により機械学習を行ったのち, そのトレーニングモデルを TFLite Converter により TFLite FlatBuffer File に変換する.

5.2 TFLite FlatBuffers

TensorFlow Lite のモデル出力形式である TFLite FlatBuffers はデータにアクセスする際にパースを要さずメモリの占有容量が小さいという特徴があり, モバイル端末や組み込みデバイスに適している. 図 6 の下側にある通り, サーバで生成された TFLite FlatBuffer File をクライアント側で TFLite Interpreter を用いることでデバイス上で利用することができる. 本研究ではこの TensorFlow Lite のバージョン 2.0.0 を用いて時系列データ学習モデルを Android 端末に組み込むことを目指す.

6. 深層学習によるトラフィック予測

4 章で述べたとおり, 時系列データの学習には LSTM をはじめとした RNN を使用するのが一般的であるが, 本研究で使用する TensorFlow Lite は RNN モデルを使用したカスタムモデルを動作させるための実装がまだサポートされていない [8]. しかしながら, ここ 1 年以内に RNN モデルの TensorFlow Lite モデルへの変換が可能になったことから, 近い将来にカスタムモデルを動作させる実装がサポートされるようになるようになることを期待しつつ, 現段階でカスタムモデルに対応済の重回帰分析での実装を進めた.

本実験では LSTM を用いない重回帰分析での学習を行い, LSTM と比較を行うことで重回帰分析を用いたモバイル端末への機械学習実装の可能性を検証した.

6.1 実験方法

本実験では, Android 端末 5 台のパケット通信時のスループットデータをサーバ上で予測した. 入力データを t -10 秒から t - 1 秒までの 10 秒間のスループットの値とし, この入力データから t 秒でのスループットの値を予測した. また, 計 181 秒間の通信のうち, 7 割を学習データ, 3 割をテストデータとして学習を行った.

6.1.1 LSTM による予測結果

図 7, 8, 9, 10, 11 は 5 台の各端末において, 学習データを入力として与え予測を行った結果である. また, グラフの青線が正解データの値で, オレンジの線が予測データの値である. どのデータも非常に高い正確度であることが分かる.

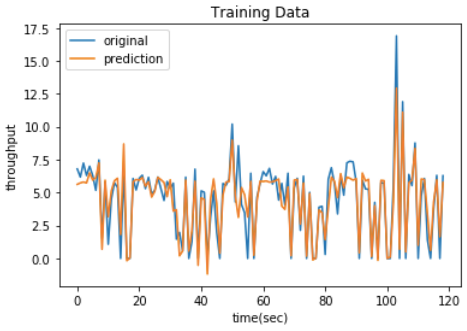

図 7 学習データによる予測 (端末 1)

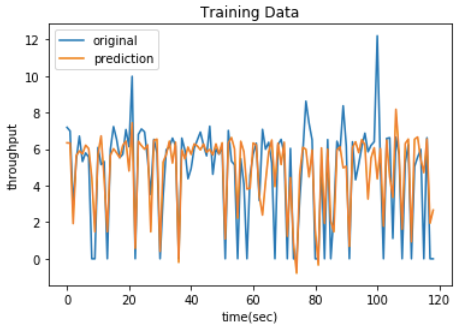

図 8　学習データによる予測 (端末 2)

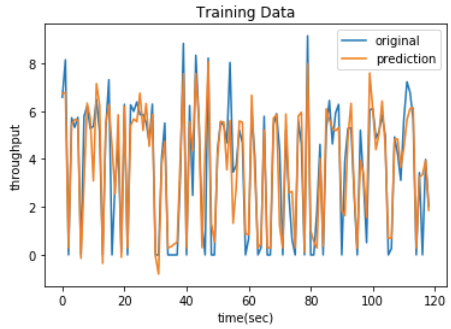

図 9　学習データによる予測 (端末 3)

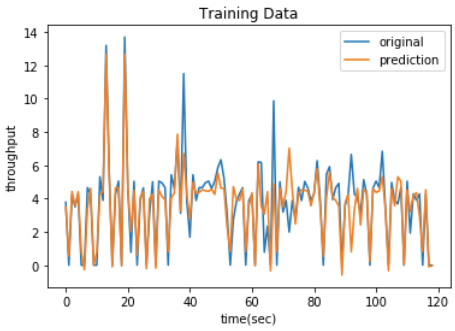

図 10　学習データによる予測 (端末 4)

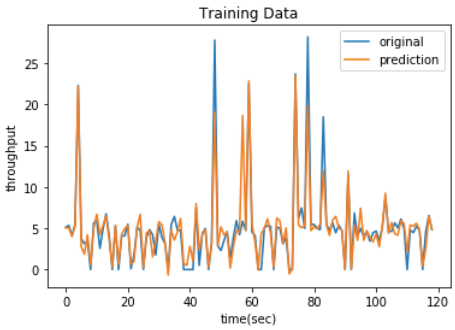

図 11　学習データによる予測 (端末 5)

図 12, 13, 14, 15, 16 はテストデータを入力として与え予測を行った結果である．また，グラフの青線が正解データの値で，オレンジの線が予測データの値である．テストデータの予測については学習データの予測に比べると正確度が劣っているが，スループットの増減のタイミングをよく捉えることができている．

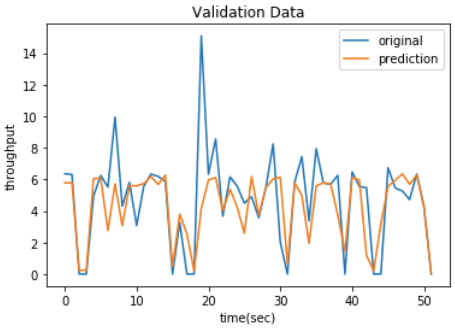

図 12　テストデータによる予測 (端末 1)

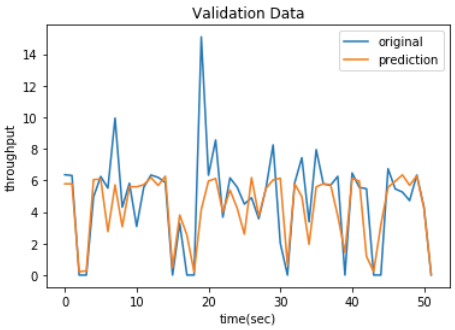

図 13　テストデータによる予測 (端末 2)

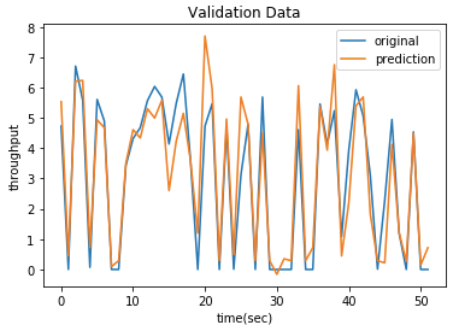

図 14　テストデータによる予測 (端末 3)

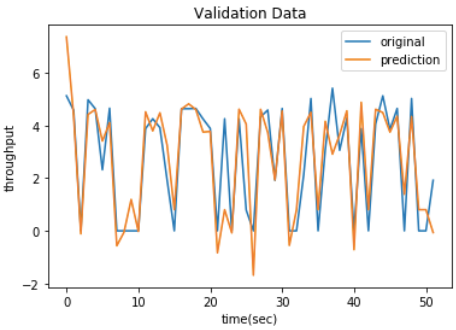

図 15　テストデータによる予測 (端末 4)

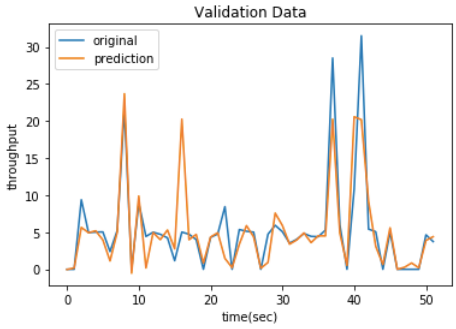

図 16　テストデータによる予測 (端末 5)

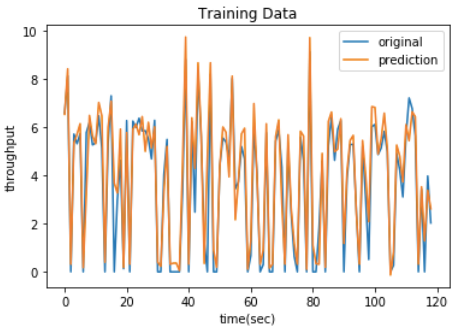

図 19　学習データによる予測 (端末 3)

6.1.2　重回帰分析による予測結果

図 17, 18, 19, 20, 21 は 5 台の各端末において, 学習データを入力として与え予測を行った結果である. また, グラフの青線が正解データの値で, オレンジの線が予測データの値である. こちらの場合も学習データによる予測は非常に高い精度の結果となった.

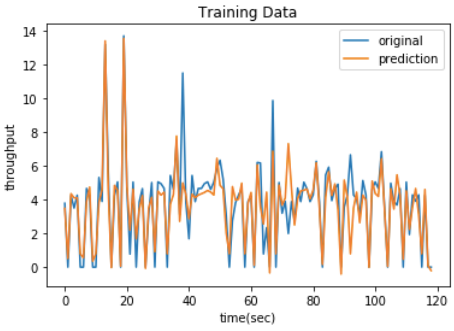

図 20　学習データによる予測 (端末 4)

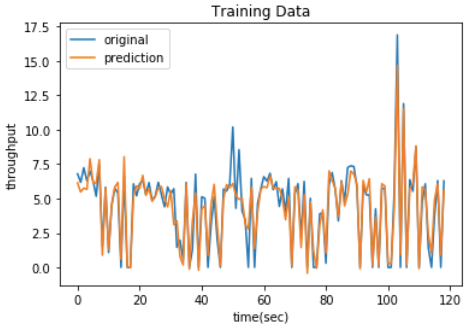

図 17　学習データによる予測 (端末 1)

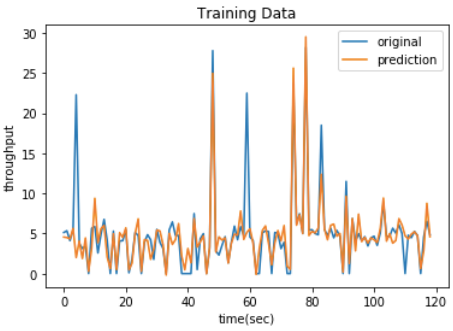

図 21　学習データによる予測 (端末 5)

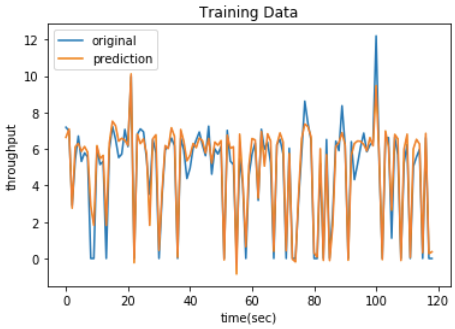

図 18　学習データによる予測 (端末 2)

図 22, 23, 24, 25, 26 はテストデータを入力として与え予測を行った結果である. また, グラフの青線が正解データの値で, オレンジの線が予測データの値である. テストデータの予測に関しても, 学習データの予測結果 には及ばないものの 2 色のグラフが一致している箇所が多く見られる結果となった. また LSTM を使用した場合の精度と比べて大きな差はなく, LSTM を使わない重回帰分析による予測であっても十分にスループットの予測が可能であることがわかった.

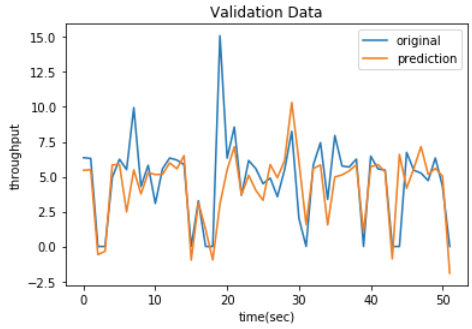

図 22　テストデータによる予測 (端末 1)

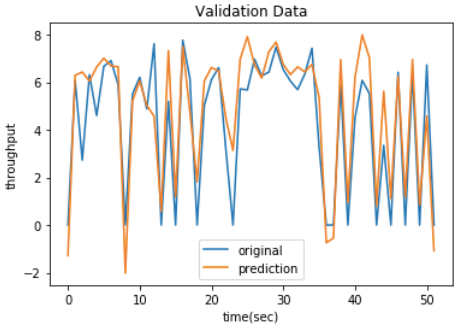

図 23　テストデータによる予測 (端末 2)

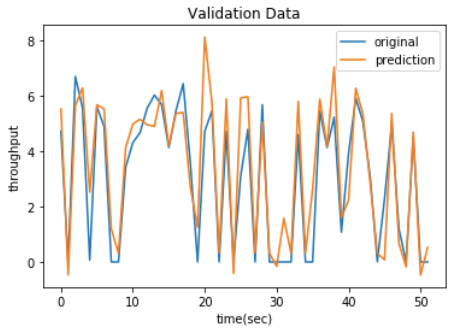

図 24　テストデータによる予測 (端末 3)

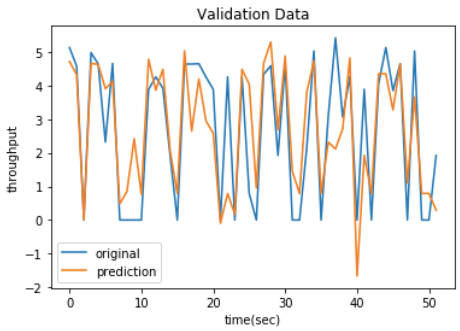

図 25　テストデータによる予測 (端末 4)

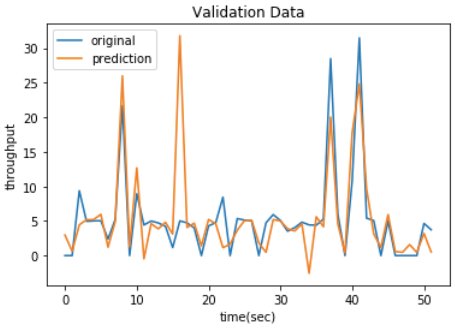

図 26　テストデータによる予測 (端末 5)

7.　端末内深層学習予測処理

前章にて端末外のサーバでのスループットの予測が可能であることが示された．本予測を端末内で行うには，当該モデルの TFLite 形式への変換，TFLite による端末内予測の実行，予測精度の向上などが重要と考えられる．

7.1　学習モデルの変換

6 章で作成した重回帰分析によるスループットの学習モデルを 5 章の流れに沿って TFLite Converter を用いて TensorFlow Lite 対応形式に変換を行い，変換が可能であることが確認された．この TensorFlow Lite 形式に変換された学習モデルにより，TFLite Interpreter を用いてサーバ上で予測を行い，その結果を変換前のモデルでの予測結果と比較した．

表 1　TFLite 変換前と変換後の予測結果の比較 (抜粋)

変換前	変換後	-3.2548912	-3.2548914
5.0201435	5.020143	4.448762	4.448762
5.5568385	5.556838	5.6119604	5.6119604
5.089124	5.0891237	1.956681	1.9566809
0.5239011	0.5239008	5.9535933	5.9535933
4.2234387	4.2234383	5.257978	5.2579784
5.083849	5.083848	6.12034	6.1203413
5.730351	5.7303505	4.363218	4.3632174
0.43309072	0.43309066	2.4153943	2.4153943

表 1 より，TensorFlow Lite 形式変換前後のモデルの予測値の差は 10 の-6 乗から 10 の-7 乗程度という非常に僅かな誤差にとどまっていることが確認できた．今後はこのモデルを図 27 のようなアプリケーションとして端末に組み込んで利用することを目指す．

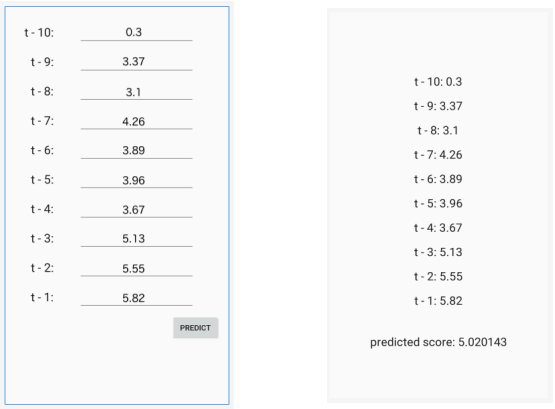

図 27 アプリケーションの入力画面, 結果出力画面のイメージ

7.2 予測精度の向上

端末のスループットデータの予測に関しては, 文献 [9] などにて, より複雑な条件下でのより高い精度を目指した予測に関する考察が行われている. これらの既存研究の成果を利用していきながらさらに高い正確度の予測が実現できると考えられる.

8. まとめと今後の課題

本研究では時系列データの予測モデルを Android 端末に導入し端末内で輻輳を事前に予知して輻輳を制御することを最終目標とし, 本稿では時系列データ予測の方法として一般的であるもののカスタムモデル未対応である LSTM を用いる代わりに, 重回帰分析を用いて端末のパケット通信時のスループットの予測を行い, LSTM を用いた予測結果との比較を行った. 過去 10 秒間のスループットデータのみを参考にしてスループットの値を予測するという非常に簡易な条件下の予測で, LSTM を用いた場合とほぼ同等の予測を行うことができるということが確認できた. また, 実験により作成したスループットデータの予測モデルを TensorFlow Lite 形式に変換を行い, 当モデルを精度を保ったまま端末内部で処理することが可能であることが確認された. これにより, 重回帰分析による時系列データ学習モデルの Android 端末への実装の可能性が示された.

今後の課題としては, 今回作成した予測モデルを Android 端末内にアプリケーションという形で組み込み端末上で利用できるようにしていきたいと考えている. また, LSTM によるカスタムモデルの導入が可能になり次第, LSTM モデルの端末での実装も行っていく予定である. さらに, これらのモデルを端末に導入した際にもサーバ機と同等の精度や処理速度での予測を行うことができるのかどうかを検証していきたいと考えている.

参考文献

[1] Gartner, https://www.gartner.com/en/newsroom/press-releases/2020-01-28-gartner-says-worldwide-smartphone-sales-will-grow-3–

[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] Android, https://developer.android.com

[6] statcounter, https://gs.statcounter.com/os-market-share/mobile/

[7] TensorFlow Lite, https://www.tensorflow.org/lite?hl=ja

[8] Unsupported operations, https://www.tensorflow.org/lite/guide/ops_compatibility#unsupported_operations (2020/05 閲覧)

[9] Yamamoto A. et al. (2019) Prediction of Traffic Congestion on Wired and Wireless Networks Using RNN. Proceedings of the 13th International Conference on Ubiquitous Information Management and Communication (IMCOM) 2019. IMCOM 2019. Advances in Intelligent Systems and Com- puting, vol 935. Springer, Cham