深層学習を用いた輻輳予測モデルの Android 端末への実装と性能評価

理学専攻・情報科学コース　2040644　佐藤 里香　（指導教員：小口正人）

1　はじめに

近年, TensorFlow Lite などのように機械学習のモデルをスマートフォン等の端末に組み込んで, 推定処理を端末内で完結させる環境が整備されつつあり, 推定処理にリアルタイム性が求められるアプリへの活用が期待されている.

本稿では, Android 端末上でトラフィックの輻輳を予測して輻輳制御を行うシステムを想定し, 予め高性能なサーバで端末におけるトラフィックデータを学習した輻輳予測モデルを, TensorFlow Lite により検証用の Android アプリに組み込み, その性能を検証し実現可能性を評価する.

2　関連研究

2.1　TensorFlow Lite

TensorFlow Lite[1] 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 により学習されたモデルを 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によりデバイス上で推論を行うことができる.

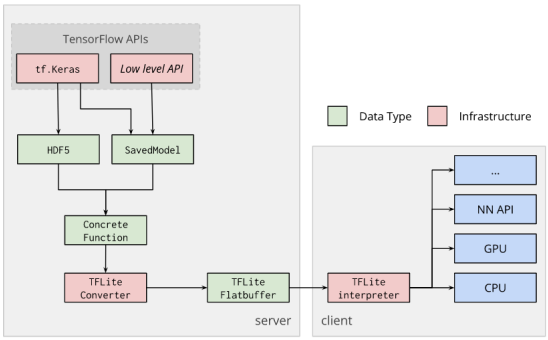

図 1: 学習モデル変換の流れ ([2] より引用)

- TFLite Converter

  TFLite Converter は機械学習モデル形式の変換を行うコンバータ (変換器) である. 図 1 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, その学習モデルを TFLite Converter により TFLite FlatBuffer File に変換する.

  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.

- TFLite Interpreter

  図 1 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 を用いることにより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 を用いて輻輳予測モデルを Android 端末に組み込み利用する.

2.2　輻輳制御ミドルウェア

本研究で取り扱うアプリ実行形態が対象とする事例の一つとして輻輳制御ミドルウェア [3](図 2) と呼ばれるツールが挙げられる. 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, 端末間で輻輳の状態を把握し, 協調して輻輳を制御することを目的としている.

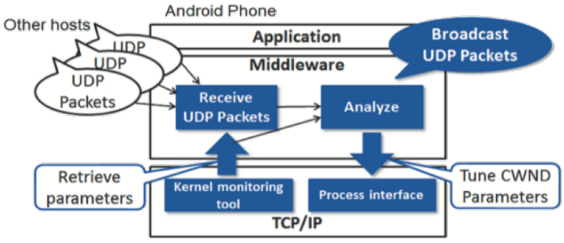

図 2: 輻輳制御ミドルウェア

このミドルウェアを用いて輻輳制御するには, 輻輳の予兆を端末側で検知することが必要となり, 過去検討では深層学習を用いた手法が提案されている [4]. そこで本稿では, その具体的な実現手段として, TensorFlow Lite を適用したモデルを端末に組み込み, 端末上で予測処理を実行させる.

本章において, 端末内予測処理検証用アプリに組み込む学習モデルを作成する. まず 2.4 においてサーバ上でトラフィックデータを LSTM により学習させる. 続いて 2.5 でそのモデルを TensorFlow Lite により端末に組み込める形式に変換する.

2.3　トラフィック予測の概要

2.2 で述べた輻輳制御システムでは, 同一アクセスポイントに接続された端末数と RTT の増減比率という 2 つのパラメータから, 適切な輻輳ウィンドウの補正値を設定している. 本研究が想定するシステムでは, 輻輳を検知して制御するのではなく, 輻輳が起こる前にその予兆を捉える必要があるため, 輻輳を示す指標として接続端末台数を採用し, その変化を予測する. また, 端末の接続台数が増えるにつれて合計スループットが大幅に減少することが先述の先行研究で明らかになっていることから, スループット値をもとに接続端末台数の多寡を予測するモデルを作成する.

2.4　TensorFlow モデルの作成

2.2 で述べた輻輳制御システムでは, 同一アクセスポイントに接続された端末台数と RTT の増減比率から適切な輻輳ウィンドウの補正値を設定している. 本研究が想定するシステムでは, 輻輳を検知して制御するのではなく輻輳発生前にその予兆を捉える必要があるため, 輻輳を示す指標として接続端末台数を採用しその変化を予測する.

本稿では, 過去 10 秒間のスループットを入力として, 同一アクセスポイント内の接続端末台数が設定した閾値に達していれば 1, 下回っていれば 0 を出力するモデルを構築する. そのモデルを作るための学習データを, Android 端末の台数を 1 台, 3 台, 5 台, 7 台, 9 台と変化させてパケット通信を行わせることで取得し, サーバ上において LSTM を用いて学習させた. また, 500 秒間の通信のうち, 400 秒を学習データ, 100 秒をテストデータとして学習を行って, 閾値を変化させて 4 パターンのモデルを作成した.

表 1: 予測における正解率

閾値	学習データ	テストデータ
3 台	100%	100%
5 台	99.5%	98.2%
7 台	95.5%	90.5%
9 台	97.5%	94.1%

表 1 は, 学習データ, テストデータのそれぞれを入力データとし, 閾値を変化させて予測させた際の正解率である. 全てのケースで正解率が 9 割を超えており, 非常に高い精度で予測ができていることがわかる.

続いて作成したモデル (TF モデル) の 1 つを 2.5 で端末に組み込める形式に変換する.

2.5 TensorFlow Lite モデルの作成

2.4 にて作成した TF モデルを, 2.1 の流れに沿って TFLite Converter を用いて TensorFlow Lite 対応形式に変換を行った.

続いてこの変換後のモデル (TFLite 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ある TF モデルによる予測結果と比較したところ, 両モデルによる予測結果は完全に一致し, モデル変換によって予測結果が変わることはないことが確認できた. この TFLite モデルをアプリとして端末に組み込む.

3 TFLite モデルの性能評価

3.1 学習モデルを組み込んだアプリケーション

2.5 にて作成した TFLite モデルを組み込んだ図 3 のような Android アプリ (検証用アプリ) を実装した. このアプリを用いて, TFLite モデルによる端末上での予測性能を, 予測精度, 予測時間, CPU 使用率で評価する.

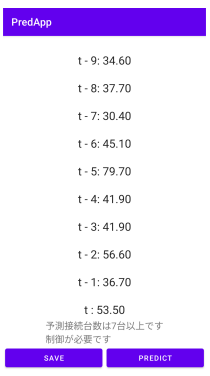

図 3: TFLite モデルを組み込んだ輻輳予測アプリ

3.2 予測精度

サーバと端末の実行環境の違いによる予測精度の変化の有無を確認するため, 両環境での TFLite モデルによる予測結果を比較したところ, 予測結果は完全に一致し, 端末上においてサーバと同じ精度での予測が可能であることがわかった.

3.3 予測時間

続いて検証用アプリで予測にかかる時間を計測したところ, 約 3.14 ミリ秒という結果になった. サーバでの予測時間は約 1.00 ミリ秒であり, 端末上での予測では 3 倍以上の時間を要しているが, 今回想定する輻輳制御 [3] によると制御の周期は 0.3 秒程度であるため, この速度差は許容範囲内であるといえる.

3.4 CPU 使用率

最後に, 検証用アプリ使用時の端末の CPU 使用率を計測したところ, 予測処理時において 7%であった. TensorFlow Lite 公式サイトが紹介するサンプルの機械学習アプリで同様の検証を行ったところ, 予測処理時における CPU 使用率は 15%となり, この結果と比較しても十分低い値であるといえる.

4 まとめと今後の課題

本研究では, Android 端末上機械学習予測処理の実現可能性を検証すべく, Android 端末上でトラフィックの輻輳を予測して制御を行うシステムを想定し, サーバで学習させた輻輳予測モデルを, TensorFlow Lite により検証用の Android アプリに組み込みその性能をサーバ上でのパフォーマンスと比較し評価した. 予測精度に関しては, サーバと同等の精度での予測が可能であること, また処理速度に関しては, サーバ上と比較すると僅かに劣るものの十分有用なものであることがわかり, 端末上輻輳予測処理の実現可能性が示された.

今後の課題としては, 本研究で目指している輻輳の事前検知を実現させるため, より早い段階での高精度な予測を実現させること, そして最終的には端末上で完結する輻輳制御システムの実現を目指していきたいと考えている.

参考文献

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

[2] TensorFlow Lite コンバータ, https://www.tensorflow.org/lite/convert?hl=ja

[3] Ai Hayakawa, Saneyasu Yamaguchi, Masato Oguchi, Reducing the TCP ACK Packet Backlog at the WLAN Access Point, Proc. ACM IMCOM2015, 5-4, January 2015

[4] 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 Computing, vol 935. Springer, Cham