修 士 学 位 論 文

題　　目

SDN マルチコントローラ環境における負荷分散を
目的とした BiLSTM による負荷予測
Load Prediction Using BiLSTM
for Load Balancing in SDN Multi-Controller Environments

指 導 教 員　　石橋勇人 教授

令 和 6 年（ 2024 年）度 修 了

（No. BGB23010 ）　　趙　麓暉

大阪公立大学 大学院
情報学研究科 学際情報学専攻

目次

図目次

表目次

1 はじめに

ソフトウェア定義ネットワーク（SDN）は,制御プレーンとデータプレーンを分離する特徴を有するネットワークであり,ネットワークの転送ルールを定義する機能をコントローラに委譲することで,従来のネットワーク管理方式を変革する.これにより,ネットワークスイッチは単純な転送デバイスとして扱われるとともに,ポリシーの適用やネットワーク構成の簡素化が実現される[1].OpenFlow 1.2 以降のプロトコルでは,SDN が OpenFlow スイッチの複数コントローラへの接続をサポートするようになった.この特性の登場および SDN におけるデータ需要の増加に伴い,SDN のコントローラクラスタ技術が注目され,研究課題として取り上げられるようになった[1].

しかしながら,マルチコントローラの配置において,スイッチとコントローラの対応関係が静的に設定されている場合,コントローラ間の負荷分散が困難となるという課題が存在する.この制約は,ネットワーク性能の低下を引き起こす可能性がある[2].したがって,この問題の解決は極めて重要である.SDN のインターフェースプロトコルとして広く採用されている OpenFlow v1.3 は,コントローラの障害時のフェイルオーバーや負荷分散の問題を適切に解決できるとされる[3].SDN のコントローラには,master,equal,slave の 3 つの役割があり,それぞれの役割は動的に変更可能である[4].ただし,スイッチの管理および設定を行う権限を有するのは,master 状態のコントローラのみである.この特性により,動的なスイッチの移行が可能となり,複数のコントローラ間の負荷を効果的に分散させる手法として有効に機能する[2].しかし,OpenFlow 規格ではスイッチの移行メカニズムが明確に規定されていない.そのため,初期の従来型スイッチの移行プロセスでは,大量の移行情報フローが発生し,システムに追加のオーバーヘッドをもたらし,移行コストを増大させることが報告されている[8].さらに,初期の従来型スイッチの移行アルゴリズムでは,過負荷状態のコントローラ[9]が管理するスイッチ群の中からランダムにスイッチを選択して移行を行っていた.しかし,この手法では,スイッチと元のコントローラ間の移行回数,移行コスト,および移行距

離が考慮されておらず,その結果,コントロールノードの追加リソース消費および移行コストの増加を引き起こし,コントローラクラスタの不安定性を助長し,負荷分散の効率を低下させる要因となっていた.これらの問題は,SDN の運用に追加の負担を与える要因となる[1][10].したがって,多くの研究では,この問題を解決するために負荷分散に関連するアルゴリズムが開発されている.また,深層学習を活用し,将来の時間帯における負荷を予測することで,より適切な負荷分散パラメータを取得し,負荷分散の効率向上を図る手法が提案されている.

本研究では,BiLSTM と LSTM モデルの比較を行った上で,[2]のデータセットを用いたテストを実施し,BiLSTMモデルに適したトラフィックパターンを分析する.

以下,第2章では,SDNにおけるトラフィック負荷分散に関する関連研究について紹介する.第 3 章では,負荷分散と負荷予測についてそれぞれ説明し,それらの関係性について議論する.第 4 章では,BiLSTM モデルを用いた負荷予測に関する実験とその結果について説明する.第 5 章では,研究のまとめと今後の課題について述べる.

2 関連研究

2.1 深層学習によるネットワークトラフィック予測：最近の進捗,分析,今後の方向性

Ons Aouedi らの研究[6]では,まず,深層学習モデルとネットワークトラフィック予測について簡単に紹介した.そして,深層学習モデルがネットワークトラフィック予測に対する適用性が分析した.代表的な DL モデルには,MLP,CNN,RNN,LSTM,GRU,Transformer,GNN などがあり,それぞれの特徴と長所・短所が分析されている.次に,これらのモデルを用いた研究が紹介され,比較が行われた.代表的なデータセット（Abilene,GÉANT,SDN）を使用し,異なる DL モデルの予測性能（MSE,MAE）を比較.最後,計算負荷の軽減,データ不足の問題への対策,リアルタイム予測の最適化が今後の重要な課題に明確にした.また,GNN を用いたより高度な空間依存関係のモデリングが期待される.

Ons Aouedi らの研究では,深層学習を活用したネットワークトラフィック予測の最新技術を包括的に整理し,課題や今後の研究方向を明確に示している.しかし,この研究はまだ,より少ない特定の異なるネットワーク環境を分析している.

2.2 遅延に敏感なアプリケーションのための機械学習によるプリエンプティブ SDN 負荷分散

Abderrahime Filalir ら[7]は,SDN マルチコントローラ環境における負荷分散の最適化手法として,機械学習を用いた予測的（Preemptive）な手法を提案し,従来のリアクティブ方式と比較しながらその有効性を示している.特に,遅延に敏感なアプリケーションに適した SDN コントロールプレーンの負荷管理手法を開発し,従来の負荷分散アルゴリズムの問題点を克服することを目的としている.

具体的に,ARIMA（自己回帰和分移動平均と LSTM（長短期記憶ネットワーク）を比較し,短期予測では ARIMA が若干優れるものの,長期予測においては LSTM が ARIMA より 55%精度が向上 することを示した.次に,負荷分散の最適化問題を非線形二値最適化問題（NP 困難）として定式化し,移行のコストと

負荷分散のバランスを最適化するために,強化学習に基づくアルゴリズム 2WSLS を提案した.

本研究では,LSTM は長期予測において高い精度を示したが,実験では単一のコントローラ負荷のみを対象としており,複数のコントローラ間の相互作用やトポロジの変化を考慮した予測が行われていない.これにより,実際の大規模ネットワークでの適用にはさらなる改善が必要である.

2.3 負荷予測に基づく SDN マルチコントローラクラスターの負荷分散戦略

Junbi Xiaora らの研究[1]では,BiLSTM+Attention モデルを用いて packet-in message の数を予測し,コントローラの過負荷を予測した際にスイッチの移行を早期決定する手法を実装し,他の手法との比較を行った.この手法により,スイッチの移行コストと負荷分散の分散が低減されることが確認された.

この方式では,スイッチ移行の最適化を行うことで,従来の手法(TSSM,DDM)と比較して負荷分散の効率を 27.6%向上させ,移行コストを 16%(TSSM 比),8%(DDM 比)削減できることを実験的に示した.また,LPSM では,スイッチ移行の対象となるコントローラを選択するために,ネットワークトラフィック,CPU 利用率,メモリ使用率,ネットワーク帯域幅などの複数の要素を考慮したアルゴリズムを導入した.

しかし,BiLSTM + Attention モデルを用いた負荷予測を行うため,コントローラごとにモデルを維持する必要がある.このため,SDN 環境の規模が拡大した場合,予測モデルの管理コストが増加し,計算リソースの消費が課題となる.

3 SDN の負荷分散と負荷予測

3.1 SDN マルチコントローラ環境における負荷分散

現在, マルチコントローラの負荷分散戦略に関する研究は, 主に集中型意思決定モデルと分散型意思決定モデルの 2 種類に分類される[2]. 第 1 のモデルでは, ルートコントローラがローカルの過負荷コントローラに対して負荷分散指令を送信し, ローカルコントローラがその操作を実行する方式である. 第 2 のモデルでは, 各コントローラがローカルで独自に負荷分散の決定を行う方式である. これらの 2 つのモデルに共通する点は, 各コントローラが分散アーキテクチャ内の一部のスイッチを担当し, さらに各コントローラ間で情報を交換することである. 従来のマルチコントローラ SDN アーキテクチャにおいて, コントローラとスイッチの接続は静的に設定されており, クラスタ内で特定のコントローラが管理するスイッチの数が増加するにつれて, コントローラの負荷も増大する[10]. さらに, コントローラのトラフィックがピークに達すると, ネットワーク全体の遅延が増加する要因となる[11].

SDN におけるマルチコントローラ負荷分散技術は, スイッチマイグレーションを用いることで実現可能である. スイッチマイグレーションでは, 一部のスイッチを過負荷状態のコントローラから, 負荷の軽い別のコントローラへ移行させることで, コントローラ間の負荷不均衡を解決する[1]. 例として図 1 に示す.

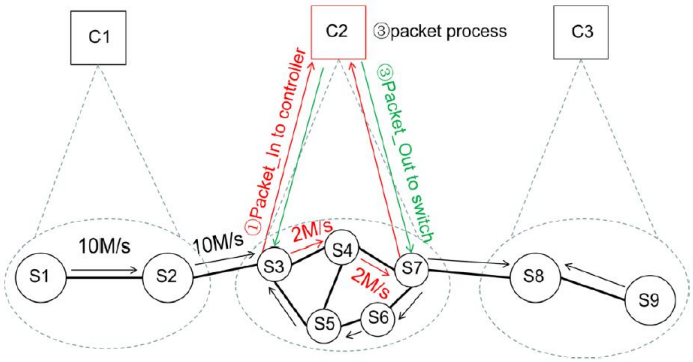

図 1 SDN における不均衡なマルチコントローラクラスター「(文献[1]より引用)」

図 1 では,3 つのコントローラ{C1, C2, C3}と 9 つのスイッチ{S1,S2,S3,S4,S5,S6,S7,S8,S9}が存在し,それぞれのコントローラが異なるサブネットを管理している.C1 は S1 および S2 の packet-in メッセージを処理し,C2 は S3, S4, S5, S6, S7 を,C3 は S8 および S9 の packet-in メッセージを処理する.packet-in メッセージがコントローラに到着すると,コントローラはメッセージを処理し,対応するスイッチに packet-out メッセージを送信する.そのため,1 つのコントローラが多数のスイッチを管理している場合,packet-in メッセージの処理量が増加し,コントローラの処理能力の限界に達する可能性がある.一般的な解決策として,過負荷状態にあるコントローラ配下のスイッチを他のコントローラへ移行させることで,負荷を分散する方法が用いられる.図 2 において,C2 が過負荷状態にあるとした場,S4 および S7 のスイッチをそれぞれ C1 および C3 へ移行させることで,C2 のリソースを解放し,負荷を軽減することができる.

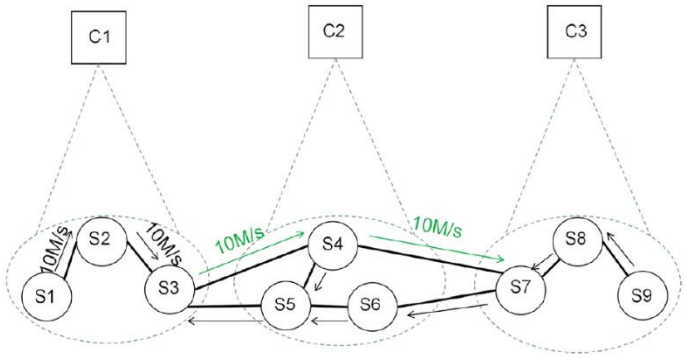

図 2 マルチコントローラクラスタによる負荷分散の取れた SDN「(文献[1]より引用)」

この負荷分散のための指標として,一定時刻の区間にコントローラがスイッチから受信する packet-in 数をコントローラーの負荷と定義する[5].

$$Load_{C_iT_n} = \sum_{S_k \in Q_{C_i}} f_{S_k T_n} \quad (1)$$

式１の意味は次の通りである. ある時間区間T_nにおいて, スイッチS_kからのリクエストメッセージの数を記録し, これを$f_{S_k}T_n$と表記する. これは, スイッチS_kがT_nにもたら負荷を示し, その総計を$Load_{C_iT_n}$と定義する

本研究では, packet-in 到達率を負荷として定義し, これに基づくトラフィック予測に焦点を当てて検討する.

3.2 トラフィックマトリックス（TM）

トラフィックマトリックス（TM）は, コンピュータネットワーク内の全ての送信元-宛先（OD）ノードの間で流れるトラフィック量を特定の時刻において表すものである. SDN において広く用いられる OpenFlow（OF）プロトコル[13]では, 各ノードの流量情報を容易に収集でき, さらに他のノードへのトラフィック量も抽出可能である. そのため, 基盤となるネットワーク構造に依存することなく, 高精度なトラフィックマトリックス（TM）を構築することができる[12].

ここでは, [2]に基づいて構築されたトラフィックマトリックスを具体的に例示する. SDN ネットワーク G は N 個のコントローラーC={C1, C2, C3, …, Cn}および K 個のスイッチ S={S1, S2, …, Sk}で構成される. N 個のネットワークドメインを管理し, すべてのコントローラは同等の処理能力を持つと仮定されている. 各ドメインにおいて, ネットワークトラフィックおよびコントローラのワークロードは動的に変化する. コントローラ Ci によって管理されるスイッチ集合をQ_{C_i}(S∈ Q_{C_i})とし, 各コントローラの管理するスイッチの総数が全体のスイッチ数に等しくなるように以下の条件が成り立つ：

$$\sum_{i=1}^{N} |Q_{C_i}| = K \quad (2)$$

ここで,最大管理スイッチ数 M を以下のように定義する.

$$M = \max\{|Q_{C_1}|, |Q_{C_2}|, \ldots, |Q_{C_N}|\} \tag{3}$$

また,W は,スライディングウィンドウのサイズを表し,トラフィックコレクションマトリックス TCM は M*W の行列として定義される.

$$TCM = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1W} \\ X_{21} & X_{22} & \cdots & X_{2W} \\ \vdots & \vdots & \ddots & \vdots \\ X_{M1} & X_{M2} & \cdots & X_{MW} \end{bmatrix} \tag{4}$$

このような手法を用いることで,研究者は TM（トラフィックマトリックス）と OpenFlow プロトコルを組み合わせて,必要な学習データを収集することが可能となる.これにより,多様なトラフィック予測モデルの研究が促進され,より高精度な予測手法の開発に貢献できる.

3.3 深層学習モデルを用いたトラフィックマトリックス（TM）の予測

3.3.1 スライディング学習ウィンドウを使ってリアルタイムで予測

ネットワークトラフィック予測とは,トラフィック予測コンポーネントが各コントローラ上で動作し,トラフィック収集コンポーネントのデータを利用して将来のデータを予測する場合に過去および現在のネットワークトラフィックデータを用いて将来のネットワークトラフィックを予測することである[2].

リアルタイムのトラフィックマトリックス予測には,継続的なデータ供給と学習が必要である.しかし,時間の経過とともにタイムスロットの総数が増加し,計算の複雑さが高まるという問題が発生する.この問題に対処するために,スライディング学習ウィンドウが必要となる.これは,直近の固定数のタイムスロットを学習し,現在のトラフィックベクトルX_tを予測する手法である[15](図 3).ここで,M はコントローラが管理するスイッチの最大数を表し,W はスライディングウィンドウのサイズを示す[2].

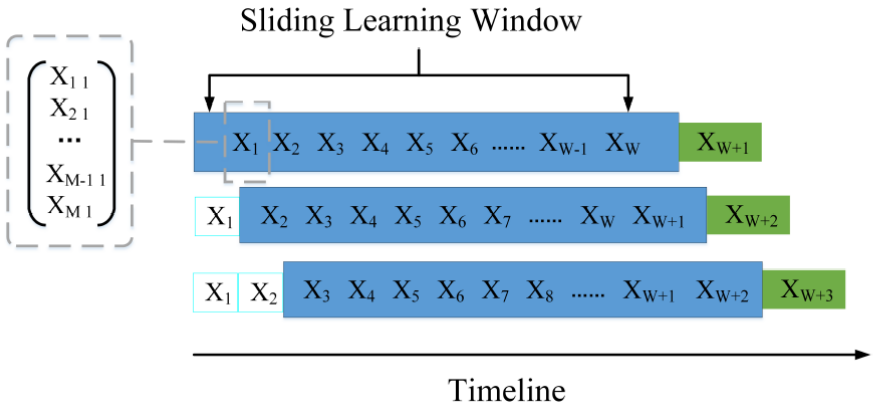

図 3 時間経過に伴うトラフィックマトリックス予測「(文献[2]より引用)」

トラフィック収集マトリックスにおいて, X_{ij}はコントローラドメイン内の i 番目のスイッチがコントローラに送信するトラフィック量を表す. 具体的には, タイムスロット$[j, j+\Delta t]$ においてスイッチがコントローラに送信した packet-in メッセージの数に相当する[2].

深層学習を用いたコントローラ負荷予測の過程は, コントローラの負荷状況を事前に把握し, 将来的にコントローラが過負荷となると判断した際に, 事前に負荷分散操作を実施することである. これにより, コントローラの高負荷によるユーザ応答時間の急激な増加という問題を解決し, ユーザエクスペリエンスの品質を向上させる[2]. TM（トラフィックマトリックス）予測は通常, 訓練と予測の 2 つの段階に分けられる.

訓練段階では, 入力層に訓練データを与え, ニューラルネットワークのパラメータを動的に調整することで, 入力データに対する期待される出力値を達成する. ニューラルネットワークは, データから学習し, 入力データと対応する目標値との関連モデルを構築する. 予測段階は, ニューラルネットワークのテスト段階であり, 新しい未見の入力データをネットワークに与え, 出力を計算して新しい入力データの結果を予測する. 訓練段階では, W 個のベクトル$(X_{t-W}, X_{t-W+1}, \ldots, X_{t-1})$を W タイムステップの入力として受け取り, 対応するラベ$(X_{t-W+1}, X_{t-W+2}, \ldots, X_t)$を用いて softmax 損失を計算し, その後

BPTT（Backpropagation Through Time）を用いてモデルの重みを更新する.予測段階では,時刻 t+1 におけるモデルは W 個ベクトル $(X_{t-W+1}, X_{t-W+2}, \ldots, X_t)$を入力とし,最後のタイムステップの出力を$X_{t+1}$として取得する[2].

3.3.2 トラフィックマトリックス（TM）予測に用いられる代表的なモデル

再帰型ニューラルネットワーク(RNN)は,人工ニューラルネットワークの一種であり,内部状態を利用して入力系列を処理することができる.この特性により,RNN はトラフィックマトリックス（TM）予測のような時系列問題に適している.しかし,RNN を勾配ベースの誤差逆伝播法（Backpropagation Through Time,BPTT）により学習する際,勾配消失問題および勾配爆発問題に直面することが知られている.この問題を解決するために,さまざまな RNN の派生モデルが提案されている.本研究では, 2 種類の LSTM 系列モデルを用いる[12].

（1）LSTM

メモリブロックと呼ばれるユニットで構成される.各メモリブロックには,情報の流れを制御するためのゲートと呼ばれる特別な乗算ユニットに加え,ネットワークの時間的状態を記憶（記憶）する自己接続を持つメモリセルが含まれる.各メモリ・ブロックには,メモリ・セルへの入力活性化の流れを制御する入力ゲート,ネットワークの残りの部分へのセルの活性化の出力の流れを制御する出力ゲート,および忘却ゲートが含まれる[15].これにより,ネットワークはデータ内の長期的な関係をより効果的に学習できる.時間ギャップに対する感度が低いため,LSTM ネットワークは単純な RNN よりもシーケンシャルデータの解析に適している.図 4 に示して[16]は,LSTM アーキテクチャおよびタイムステップ t におけるデータフローを確認できる.

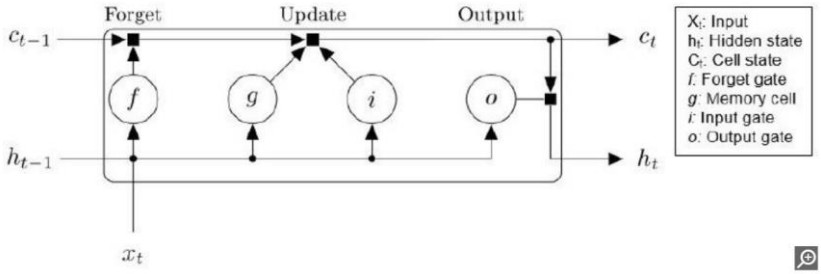

図 4 LSTM ユニットのタイムステップ t におけるデータフロー「(文献[16]より引用)」

入力ゲートへの重みおよびバイアスにより,新しい値が LSTM ユニットに入る度合いを制御する.同様に,忘却ゲートと出力ゲートへの重みおよびバイアスにより,ユニット内に値が留まる度合いと,LSTM ブロックの出力活性化を計算するためにユニット内の値が使用される度合いをそれぞれ制御する.

(2) 双方向 LSTM (BiLSTM)

従来の LSTM を拡張したもの（双方向 RNN のアイデアに基づく）で,モデルの性能を向上させることができる.BiLSTM は,2 つの LSTM を 2 つの反対方向の入力に対して組み合わせたものである.順方向と逆方向の両方のシーケンスを考慮する.これは,より良い予測のためにネットワークに追加情報を提供することができる[12].順方向 LSTM は,最初のタイムステップから最後のタイムステップに向かって演算する.逆方向 LSTM は,最後のタイムステップから最初のタイムステップに向かって演算する.2 つの LSTM コンポーネントにデータを通過させた後,演算により,チャネル次元に沿って出力を連結する.図 5 に示すようにする.

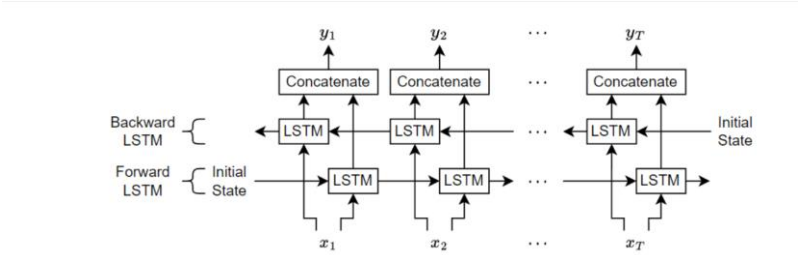

図 5 複数のタイムステップをもつ BiLSTM のアーキテクチャ「(文献[16]より引用)」

3.3.3 評価指標

平均二乗誤差（Mean Squared Error, MSE）は,予測値と実際の値との差の二乗平均を計算する指標であり,大きな誤差をより強く罰するため,外れ値に対して敏感である[6].

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y_i} - y_i)^2 \tag{5}$$

二乗平均平方根誤差（Root Mean Squared Error, RMSE） は,平均二乗誤差（MSE） の平方根であり,予測誤差の標準偏差を表す.RMSE は予測値と同じ単位で表されるため,解釈が容易である[6].

$$RMSE = \sqrt{MSE} \tag{6}$$

平均絶対誤差（Mean Absolute Error, MAE）は,予測値と実際の値の誤差の平均的な大きさを測定する指標であり,誤差の方向性を考慮せず,平均的な予測誤差を直接的に示す[6].

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i| \tag{7}$$

決定係数（Coefficient of Determination, R^2）は,従属変数（実測値）の分散のうち,独立変数（予測値）から予測可能な部分の割合を測定する指標である.R^2の値は0から1の範囲を取り,1は完全な予測を示します.R^2は,モデルが観測データにどの程度適合しているかを評価する際に有用であり,特にモデルの全体的な適合度を評価する際に役立つ[6].

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{N}(\hat{y}_i - \bar{y})^2} \tag{8}$$

4 実験と評価

4.1 BiLSTM モデルの訓練と予測

既存の公開データセットとして,GÉANT と文献[12]の SDN ネットワークのテストベッドで TM トラフィックを収集したトラフィックマトリックスデータセットがある.GÉANT[17]ネットワークは,ヨーロッパの国立研究教育ネットワーク（NREN）を相互接続するパンヨーロピアンなデータネットワークである.このデータセットは,データプレーンのネットワークトラフィックデータとして測定している.フローエントリが存在しない場合など,特定の条件下で Packet-in に関する情報を推定することは可能であるが,関連するデータを抽出するためには特徴量エンジニアリングが必要である.

文献[2]のデータセットは,OpenFlow 通信環境をシミュレートし,特徴量エンジニアリングを通じて Packet-in 到達率を表す関連データを抽出している.したがって,本実験では[2]のデータセットを利用する.

4.1.1 実験

実験環境

本実験は Google Colab 上で実施した.Google Colab は,クラウドベースの Jupyter Notebook 環境を提供し,GPU を利用できるため,ディープラーニングモデルの訓練に適している.ハードウェア構成は,GPU (NVIDIA Tesla T4) を使用し,RAM は約 12GB,ディスク容量は約 100GB（クラウド環境)である.

データセットとしては, 文献[2]の公開データセット data.csv を使用した.このデータはすで特徴量エンジニアリングが施されたものである.

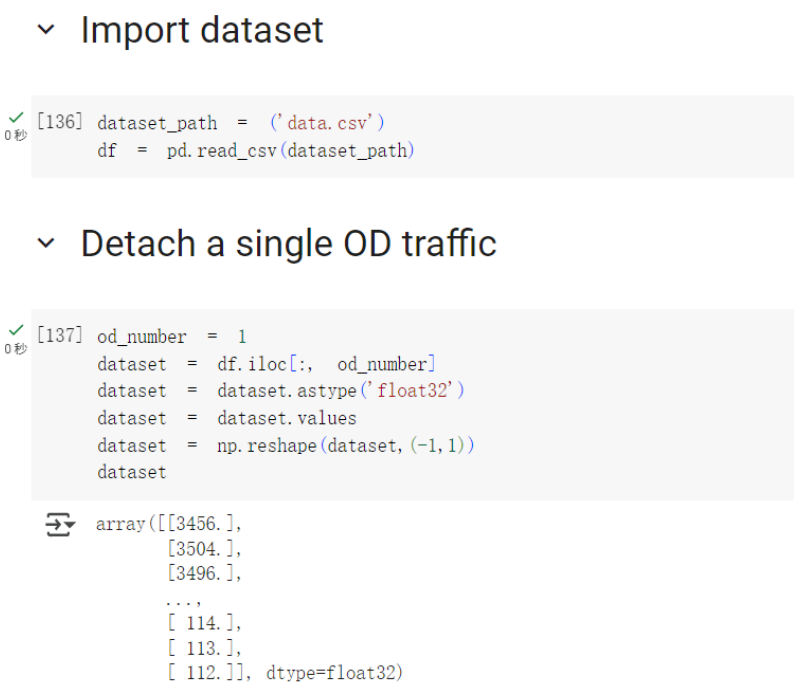

図 6 データセットの取得

データの前処理では,データの正規化には MinMaxScaler を使用し,0 から 1 の範囲にスケーリングを行った.データは 80%を訓練データ,20%をテストデータとして分割した.時系列データを扱うため,適切なウィンドウサイズ look_back を設定し,モデルへの入力データを作成した.データ形状を BiLSTM に適した形[サンプル数,タイムステップ数,特徴量数]にリシェイプした.具体的なコードを図 7 に示す.

```
[186] scaler = MinMaxScaler()
      dataset = scaler.fit_transform(dataset)

      train_size = int(len(dataset) * 0.8)
      test_size = len(dataset) - train_size
      train, test = dataset[0:train_size,:], dataset[train_size:len(dataset),:]

      def create_dataset(dataset, look_back=1):
              dataX, dataY = [], []
              for i in range(len(dataset)-look_back-1):
                  a = dataset[i:(i+look_back), 0]
                  dataX.append(a)
                  dataY.append(dataset[i + look_back, 0])
              return np.array(dataX), np.array(dataY)


      look_back = 6
      trainX, trainY = create_dataset(train, look_back)
      testX, testY = create_dataset(test, look_back)

      trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
      testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]))
      print("train_data_size: " + str(trainX.shape[0]), " test_data_size: " + str(testX.shape[0]))

train_data_size: 4793  test_data_size: 1193
```

図 7 データの前処理

モデルの訓練には,以下の 3 つの重要なコールバックを使用した.

(1) EarlyStopping とは val_loss を監視し,10 エポック連続で改善が見られない場合に訓練を停止し,最良の重みを復元する.

(2) ModelCheckpoint とは val_loss が最小値を更新するたびにモデルの重みを best_model.h5 に保存する.

(3)ReduceLROnPlateau とは val_loss が 5 エポック改善しない場合,学習率を 50% 減少させる.モデルの訓練を図 8 に示す.

```
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
checkpoint = ModelCheckpoint("best_model.h5", monitor='val_loss', save_best_only=True, mode='min')
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5, min_lr=1e-6)

history = Bilstm_model.fit(trainX, trainY,
                           validation_data=(testX, testY),
                           epochs=200, batch_size=32,
                           verbose=2, callbacks=[early_stopping])

plt.figure(figsize=(8,5))
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.title('Training & Validation Loss Curve')
plt.show()
```

図 8 モデルの訓練

これによって,BiLSTM モデルを訓練し,損失値（loss）の変化を記録した.エポック数は最大 200,バッチサイズは 32 に設定し,学習が早期停止(early_stopping)の条件を満たした場合は途中で停止する.ウィンドウサイズ look_back = (1,2,3,4,…,10)テスト中で最適な数値はサイズは 8,epoch=103,最小検証損失: $3.82e^{-5}$と訓練損失と検証損失の推移を可視化する.結果を図 9 に示す.

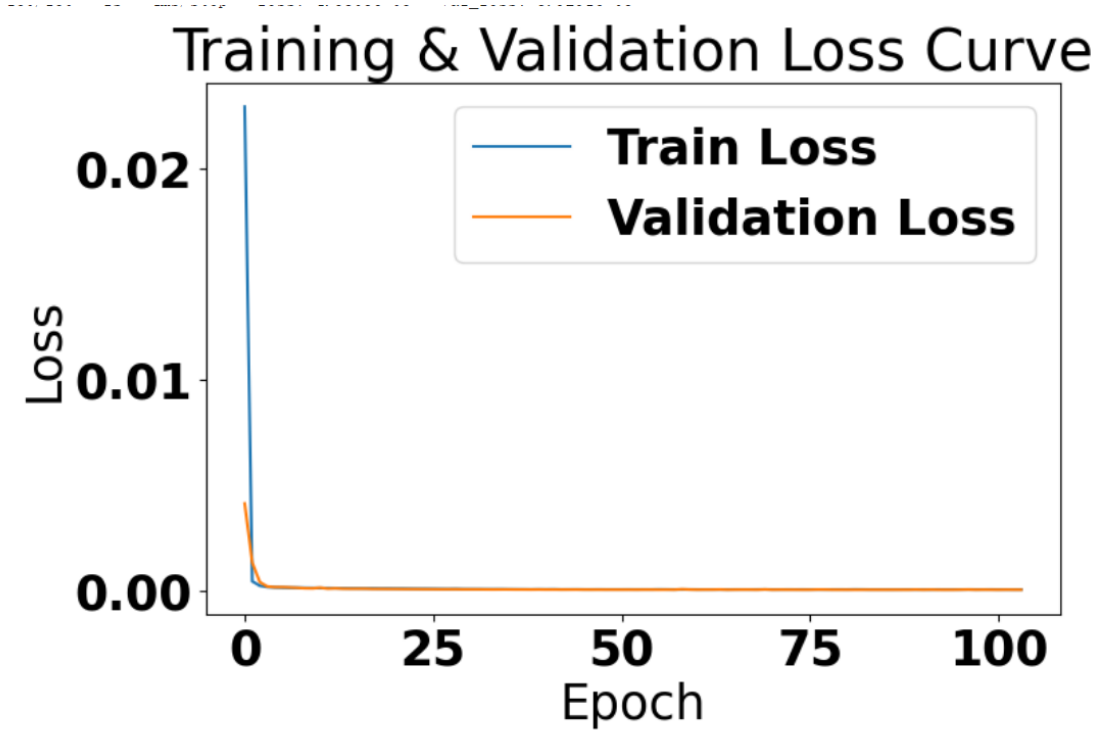

図 9 訓練損失（Train Loss）と検証損失（Validation Loss）の推移

ログを分析すると,エポック数の増加に伴い,訓練損失と検証損失の両方が減少し,その後安定していることが確認できる.具体的には,エポック 9 で検証損失が 0.00011246 と記録され,その後も損失は徐々に減少し,エポック 103 で EarlyStopping が発動した時点で,最良の検証損失は 0.00003828 となっている.このような損失の減少傾向は,モデルがデータのパターンを効果的に捉えていることを示している.学習曲線の観察から,訓練損失と検証損失の間に大きな乖離がないため,過学習の兆候は見られない.また,損失が高止まりすることなく減少していることから,未学習の兆候も見受けられない.この結果から,現在のハイパーパラメータ設定は適切であり,モデルは効果的に学習していると判断できる.

BiLSTM モデルがデータに適切に適合していることが図 10 に示されている. 横軸（X 軸）は時間ステップを示し,SDN コントローラが処理する負荷の時系列変化を表している.縦軸（Y 軸）はコントローラの負荷を表し,主に Packet-in メッセージの到達率に基づいてコントローラの負荷がどのように

変動するかを示している結果からわかるように， BiLSTM は過去と未来の両方の情報を利用して予測するため，特にデータセットの終盤において LSTM よりも誤差が小さくなる．このことが BiLSTM の優位性を示している．

LSTM は逐次的な処理のため，1000 ステップ以降のデータを学習時に直接使えない．BiLSTM は双方向の情報を持つため 1000 ステップ以降の変化を予測時に考慮できる．これにより，LSTM の場合は「過去データのみに基づいて減少傾向を推測」するが，BiLSTM は「すでに減少することを知っている」ように振る舞い，より実測値に近づく．結果の一部を拡大したグラフを図 11 に示す．

また，1000 タイムステップ付近では，急激な負荷の低下が発生している．LSTM は直前のデータ（まだ負荷が高い状態）に影響され，減少スピードが遅くなる傾向がある．BiLSTM はすでに減少のパターンを把握し，先回りして予測を最適化できる．LSTM は長期記憶が弱いため，終盤になるほど誤差が増大．BiLSTM は双方向に情報を流すため，終盤の負荷パターンにも過去データが影響しやすく，滑らかな予測曲線を描ける．

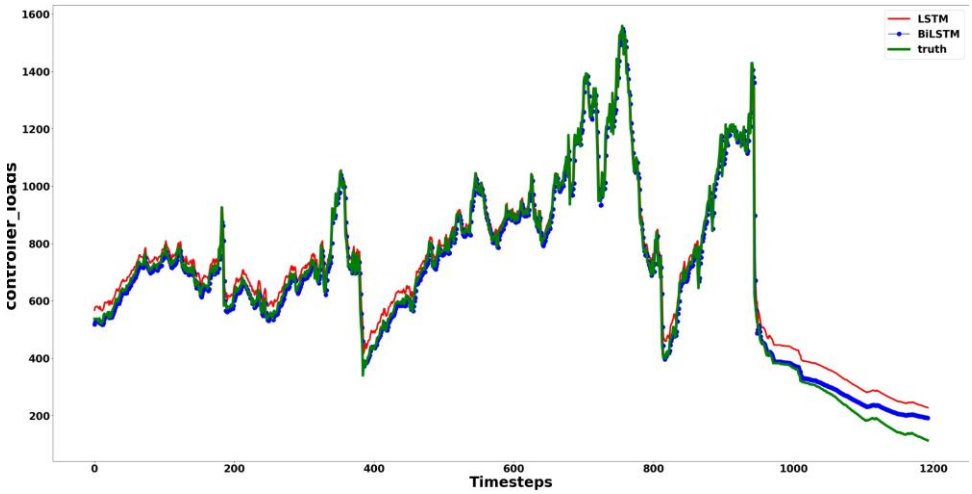

図 10 20%テストデータセットのデータフィッティング

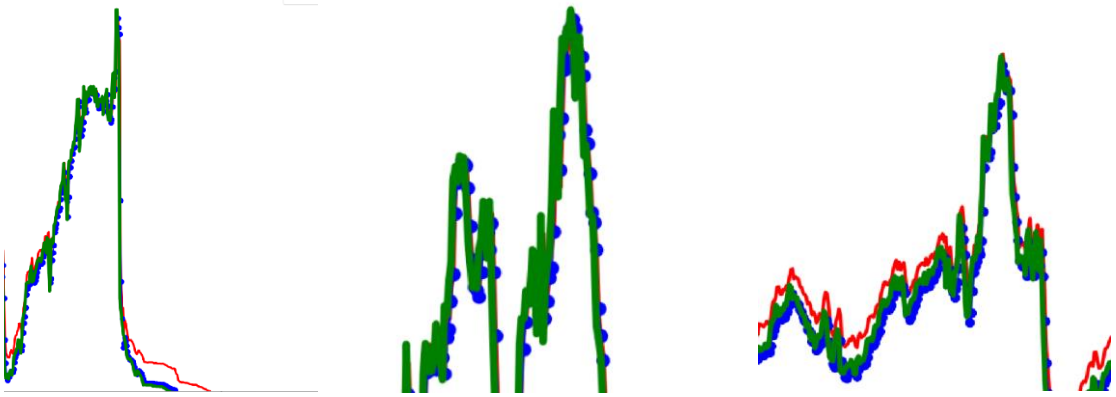

図 11　図 10 のステップ 800–1000 以降, 600–800, 200–400 の部分拡大

文献[2]のモデル(LSTM)と比較した結果を表 1 に示す.

表 1 LSTM と BiLST の結果比較

モデル	MSE(10^{-5})	RMSE(10^{-3})	MAE(10^{-3})	R^2	look_back
LSTM	19.5	13.9	11.6	0.92	1
BiLSTM	11.47	10.7	6.7	0.95	1
LSTM	17.2	13.14	11.03	0.93	2
BiLSTM	12.9	11.36	7.81	0.95	2
LSTM	24.5	15.6	13.16	0.9	3
BiLSTM	11.63	10.7	7.69	0.96	3
LSTM	41.4	20.3	17.3	0.84	4
BiLSTM	6.15	7.84	5.12	0.97	4
LSTM	38.7	19.5	16.9	0.85	5
BiLSTM	5.29	7.27	4.77	0.97	5
LSTM	39.7	20	16.7	0.87	6
BiLSTM	5.03	7.09	4.39	0.98	6
LSTM	15.18	12.32	9.83	0.94	7
BiLSTM	5.95	7.71	5.12	0.97	7
LSTM	26.9	16.4	14.9	0.89	8
BiLSTM	3.71	6.09	3.1	0.98	8
LSTM	6.10	7.81	4.61	0.97	9
BiLSTM	6.54	8.08	5.30	0.97	9
LSTM	10.15	10.07	6.97	0.96	10
BiLSTM	5.17	7.19	4.61	0.98	10

本研究では,BiLSTM を用いた SDN ネットワーク負荷予測モデルを構築し,その性能を評価した.モデルパラメータの観点から比較的良好な予測精度を示した.

具体的に,look_back=8 ではBiLSTMの特徴(双方向から情報を捉えられる)が上手く働き,大幅に誤差が下がった.look_back=9 ではわずかな長さの違いによりモデルの複雑さや学習の安定性が変化し,結果的に LSTM が若干良い結果となった.いずれにしても,R^2の値はほぼ同等であり,BiLSTM も LSTM も高い予測性能を示していることから,モデルやデータに応じた細かいチューニングやパラメータ調整がキーとなる.これらの要因により,最終的に「look_back が 8 のときは BiLSTM が最良,9 のときはLSTMのほうがわずかに良い」という結果が得られたと考えられる.

5. おわりに

本論文では,OpenFlow に基づく通信特徴抽出を用いて,Packet-In 到達率をシミュレートしたデータセットで BiLSTM モデルの訓練を行い,その結果を従来のモデルと比較した.また,トラフィックマトリックスによるトラフィック予測手法の改良について以下の考察を行った.具体的には,[2]のデータセットを用いて BiLSTM モデルを訓練し,コールバックを活用してモデルの最適なパラメータを発見・分析し,微調整を行った.

本提案手法は,既存のテストデータセットを用いて LSTM モデルと BiLSTM モデルを比較することに留まっている.今後は,より多様なモデルと比較を行い,その性能差や適用可能性を明確にすることで,BiLSTM モデルの有効性をより厳密に検証する必要がある.

今後の課題として,本研究はアルゴリズム上の改良を実現したものの,実際のアルゴリズムと組み合わせた実地テストは行っておらず,BiLSTMモデルがスイッチ移行アルゴリズムの向上に寄与するかは未だ確定していない.また,データセットの検証には,より大規模な実験を通じてデータを収集し,特徴を抽出することで,これらのデータを活用してより正確なモデル訓練を行う必要がある.さらに,仮想環境で生成したデータセットの多様性と現実性を向上させるため,異なるネットワークトポロジーやトラフィックパターンをシミュレートすることが重要である.これにより,モデルの汎化性能が向上し,実際のネットワーク環境への適用可能性が高まる.

謝辞

本論文の完成にあたり, 石橋教授の貴重なご指導とご教示に心より感謝申し上げます. 先生の深い専門知識と的確なアドバイスは, 私の研究に大きな助けとなりました. また, 研究室の皆様には, 研究および語学面でのご協力をいただき, 誠にありがとうございました. 最後に, これまで私を支えてくださったすべての方々に, 心から感謝の意を表します.

令和6年3月4日

参考文献

[1] Xiao, J., Pan, X., Liu, J., Wang, J., Zhang, P., & Abualigah, L. (2024). Load balancing strategy for SDN multi-controller clusters based on load prediction. The Journal of Supercomputing, 80(4), 5136-5162.
[2] Zhong, H., Xu, J., Cui, J., Sun, X., Gu, C., & Liu, L. (2022). Prediction-based dual-weight switch migration scheme for SDN load balancing. Computer Networks, 205, 108749.
[3] O.N.Foundation, OpenFlow Switch Specification Version 1.3.3 (Protocol Version0x04),https://opennetworking.org/wpcontent/uploads/2014/10/openflow-spec-v1.3.3.pdf
[4] Oktian, Y. E., Lee, S., Lee, H., & Lam, J. (2017). Distributed SDN controller system: A survey on design choice. computer networks, 121, 100-111.
[5] Cui, J., Lu, Q., Zhong, H., Tian, M., & Liu, L. (2018). A load-balancing mechanism for distributed SDN control plane using response time. IEEE transactions on network and service management, 15(4), 1197-1206.
[6] Aouedi, O., Le, V. A., Piamrat, K., & Ji, Y. (2024). Deep Learning on Network Traffic Prediction: Recent Advances, Analysis, and Future Directions. ACM Computing Surveys.
[7] Filali, A., Mlika, Z., Cherkaoui, S., & Kobbane, A. (2020). Preemptive SDN load balancing with machine learning for delay sensitive applications. IEEE Transactions on Vehicular Technology, 69(12), 15947-15963.
[8] Xu, Y., Cello, M., Wang, I. C., Walid, A., Wilfong, G., Wen, C. H. P., ... & Chao, H. J. (2019). Dynamic switch migration in distributed software-defined networks to achieve controller load balance. IEEE Journal on Selected Areas in Communications, 37(3), 515-529.
[9] Cheng, G., Chen, H., Wang, Z., & Chen, S. (2015, May). DHA: Distributed decisions on the switch migration toward a scalable SDN control plane. In 2015 IFIP Networking Conference (IFIP Networking) (pp. 1-9). IEEE.
[10] Zhong, H., Fan, J., Cui, J., Xu, Y., & Liu, L. (2021). Assessing Profit of Prediction for SDN controllers load balancing. Computer Networks, 191, 107991.
[11] Zhou, Y., Ren, B., Xie, J., Luo, L., Guo, D., & Zhou, X. (2023). Enable the proactively load-balanced control plane for SDN via intelligent switch-to-controller selection strategy. Computer Networks, 233, 109867.
[12] Le, D. H., Tran, H. A., Souihi, S., & Mellouk, A. (2021, June). An ai-based traffic matrix prediction solution for software-defined network. In ICC 2021-IEEE International Conference on Communications (pp. 1-6). IEEE.
[13] Hu, F., Hao, Q., & Bao, K. (2014). A survey on software-defined network and openflow: From concept to implementation. IEEE Communications Surveys & Tutorials, 16(4), 2181-2206.
[14] Tune, P., Roughan, M., Haddadi, H., & Bonaventure, O. (2013). Internet traffic matrices: A primer. Recent Advances in Networking, 1, 1-56.
[15] Azzouni, A., & Pujolle, G. (2018, April). NeuTM: A neural network-based

framework for traffic matrix prediction in SDN. In NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium (pp. 1-5). IEEE.
[16] MathWorks：長・短期記憶（LSTM）とは，MathWorks（オンライン），入手先〈https://jp.mathworks.com/discovery/lstm.html〉（参照 2025-02-06）
[17] GÉANT Association：GÉANT Network，GÉANT Association（オンライン），入手先〈https://network.geant.org/〉（参照 2025-02-06）．