豊田工業大学　課題研究報告書

系列学習による車両走行シミュレーションのモデリング

平成 30 年 2 月

工学部　先端工学基礎学科
知能数理研究室
15083　　牧野 晃平

目 次

図 目 次

表 目 次

第1章　はじめに

1.1　研究背景

近年の自動車産業では，とても短いサイクルで新しい技術の開発が行われている．開発の中で重要な役割を果たしているのがシミュレーション技術である．シミュレーション技術は現実世界の事象を計算機上でモデリングして再現し，実際に試作等を行う前に実験を行うことで，開発サイクルの高速化および低コスト化を図るための技術である．

一般的にシミュレーションは物理モデルに沿って計算を行うことで行われる．計算機では連続値を扱うことができないため，計算機上でステップ毎に順次計算を行っていくことで，現実で行われる実験の系列全体を再現する．また，車両の三次元モデルなどの計算される対象も分割して扱い，要素毎に物理的な計算が行われる．その特性上，膨大な計算量になり，計算に掛かる実時間も多くなる．しかし，実際に試作等を行うよりはシミュレーションを行ったほうが遥かに効率的であるため，実際の現場では例え時間が掛かろうとシミュレーションを行って実験を行い，裏付けをしてから試作等に進むという過程で開発が行われている．

この計算量を削減するために近年では機械学習によるアプローチが行われるようになってきている．特に流体力学の分野では顕著で，Ladický らのランダムフォレストを用いた手法 [1] や，Yang らのニューラルネットワークを用いた手法 [2] のような研究が行われている．また，機械学習の分野では，データ駆動の手法が用いられることから大量のデータが必要となる．この問題を改善するための教師なし学習を用いたシミュレーションの手法 [3] がある．しかし，これらの手法では実際に製品開発を行うに当たって，実用に用いることができるような精度を得ることはできていない．

1.2　本研究の目的

本研究で機械学習を用いたシミュレーションの精度を向上することを目的とする．実際の開発現場では，機械学習を用いたシミュレーションは精度があまり高くないため，従来の物理シミュレータを用いた手法が標準的に用いられている．機械学習を用いたシミュレーションを実用にするためには精度向上が必須である．

車両走行シミュレーションは，タイヤの摩擦係数などのパラメタの最適化を行うために用いられる．しかし，最適化するパラメタの次元数が多いことが多く，組み合わせの数が膨大となってしまう．このような問題の中，実際の物理シミュレーションを行うと現実時間で解くことが難しくなるが，機械学習を用いたシミュレーションでは精度が物理シミュレーションほど良くない．そこで本研究は，物理シミュレーションに入力するためのパラメタの組み合わせを機械学習を用い

たシミュレーションで絞ることで，実際に掛かる時間を短縮することに用いる，というのが本研究の位置づけである．

前述したように，シミュレーションはひと続きの系列データを計算によって求めることを行っている．コンピュータビジョンの分野では，系列データの予測を行うに際して，最先端の結果を得るために，Recurrent Neural Network (RNN)（2.3 節で説明）というニューラルネットワーク構造を用いた手法 [4] がよく用いられる．だが，Yang らの手法 [2] や Tompson らの手法 [3] では，ひと続きの系列データを予測しているが，RNN のような構造が用いられていない．そこで本論文では，機械学習を用いたシミュレーションに系列モデルである RNN を取り入れてシミュレータを模倣し，精度を向上を図る．

本研究の貢献は以下の点が挙げられる．

- 機械学習を用いたシミュレーションへの系列モデルの導入

  隠れ状態を持たないモデルと系列モデルを比較検証を行うことで精度が向上したことを確認した．シミュレーションに対する系列学習の有効性を検証した．

- 複数の構造のモデルに関する調査

  一つ前の状態を入力するかどうかや，スキップコネクションの追加など，種々の構造について調査を行った．

1.3 本論文の構成

本論文は 5 章で構成されている．第 2 章では本研究に関連する研究やベースラインとしているモデルについて述べる．第 3 章では本研究が提案する系列学習による車両走行シミュレーションおよびそれに付随するモデルの提案について述べる．第 4 章では提案手法による実験結果について述べる．第 5 章では本研究のまとめと今後の課題について述べる．

第2章　関連研究

本章では関連する研究について述べる．2.1 節では本研究で対象とする NCAPFishhook 試験について述べる．2.2 節では本研究で用いるモデルであるニューラルネットワークについて述べる．2.3 節では系列モデルである RNN について述べる．2.4 節ではモデルの最適化手法について述べる．3.4 節ではスキップコネクション，2.7 節ではマルチタスク学習，2.8 節では本研究の対象に対して行われていた先行研究について述べる．

2.1　NCAPFishhook 試験

New Car Assessment Program Fishhook 試験 (NCAPFishhook 試験) [5] は National Highway Traffic Safety Administration (NHTSA) が定めた，車両の動的な転倒傾向の調査のための試験である．この試験は 2 つの手順からなる．大まかな流れとしては，第一ステップで車両が釣り針型に走行する際に入力するステアリング角を得るための予備実験を行う．第二ステップでは車両を釣り針型に走行させて，そのときの挙動から転倒傾向を確かめる．以下で詳しく説明する．

1. 入力ステアリング角の規定

   対象車両の横方向の力学的特性を確かめるための手順である．一定速度で走行している状態で準静的にステアリングを入力し，横方向加速度が $0.3g$ となるときのステアリング角を求める．ただし g は重力加速度である．

   実際の条件について述べる．50 マイル毎時で直進している状態で速度を維持する．その状態でステアリングを 0 度から 270 度まで 13.5 度毎秒の角速度で入力する．270 度まで到達した後は 2 秒間維持してこの手順は終了である．測定された横加速度の値に対して線形回帰を行い，横加速度が $0.3g$ となるときのステアリング角を δ とする．

2. 釣り針型の経路で走行

   この手順は先の手順 1 で求めた δ を用いて行われる．この手順の挙動で　NCAPFishhook 試験の結果が得られる．行う操作を述べる．この手順は一定速度で行われる．まず，規定された速度に達するまで直進しながら徐々に加速し，その速度を維持する．その速度を保ちながら図 2 のようにステアリングを入力する．ステアリングの入力の規則は，δ を入力した後，角加速度が 0 となったときにカウンターステア $-\delta$ を入力し，その角度で 3 秒維持する．その時点で手順は終了となるが，その後は徐々にステアリングを 0 度に戻す．このように入力することで釣り針型に走行する．この走行時に車輪が浮いたかどうかや転覆するかどうかなどの状態を観測する．速度については，35，40，45，47.5，50 マイル毎秒の間で実験し，安

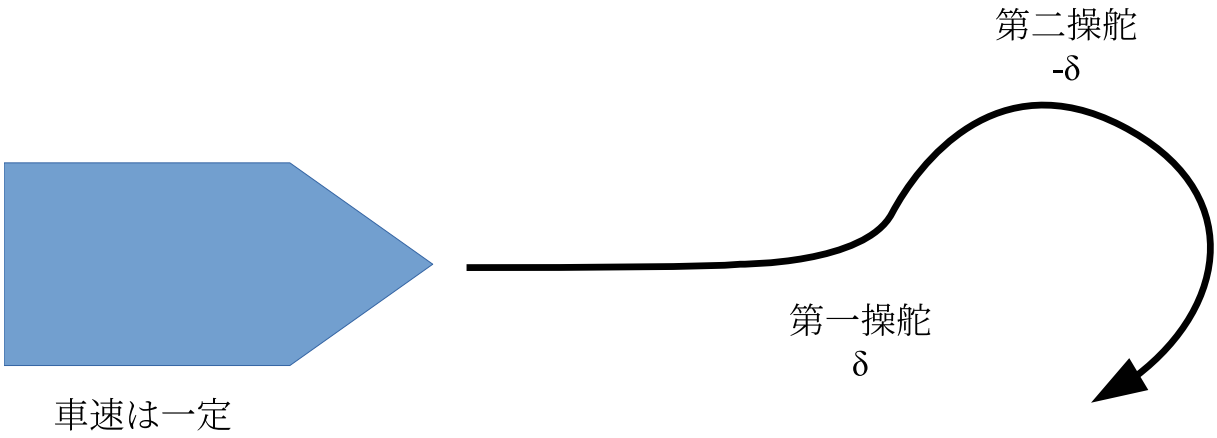

図 1: NCAPFishhook 試験における手順 2 の釣り針型に走行する様子

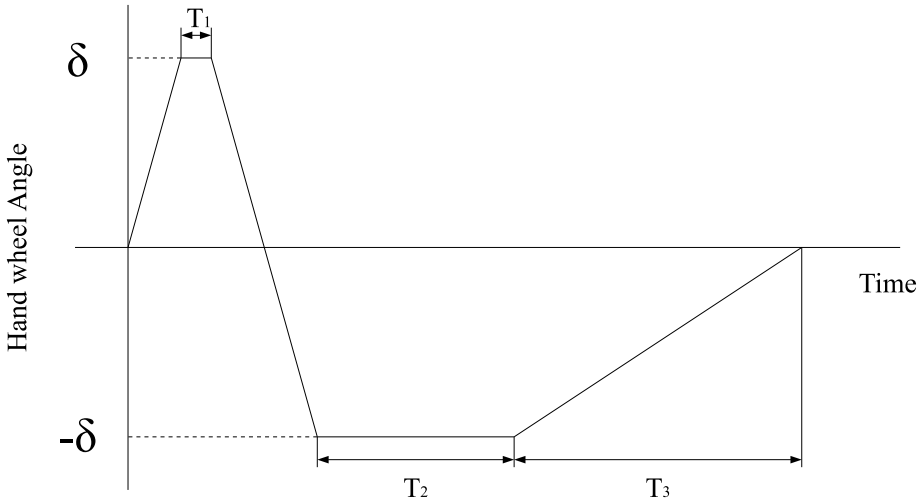

図 2: ステアリングの入力角.

定して走行できていれば速くして実験をもう一度行う．本研究の取得したデータについては後述する．走行の様子を図 1 に示す.

2.2 ニューラルネットワーク

ニューラルネットワーク [6] は人間の脳の神経細胞を模した数理モデルである．主要なモデルの一つとして，単純パーセプトロンというモデルがある．このモデルはバイアス項 b を加えると線形回帰と同一のものとしてみなすことができる．入力 $\boldsymbol{x}$ および重み $\boldsymbol{w}$ を，

$$\boldsymbol{x} = (x_0, x_1, x_2, \cdots, x_n)^\top \tag{2.1}$$

$$\boldsymbol{w} = (w_0, w_1, w_2, \cdots, w_n)^\top \tag{2.2}$$

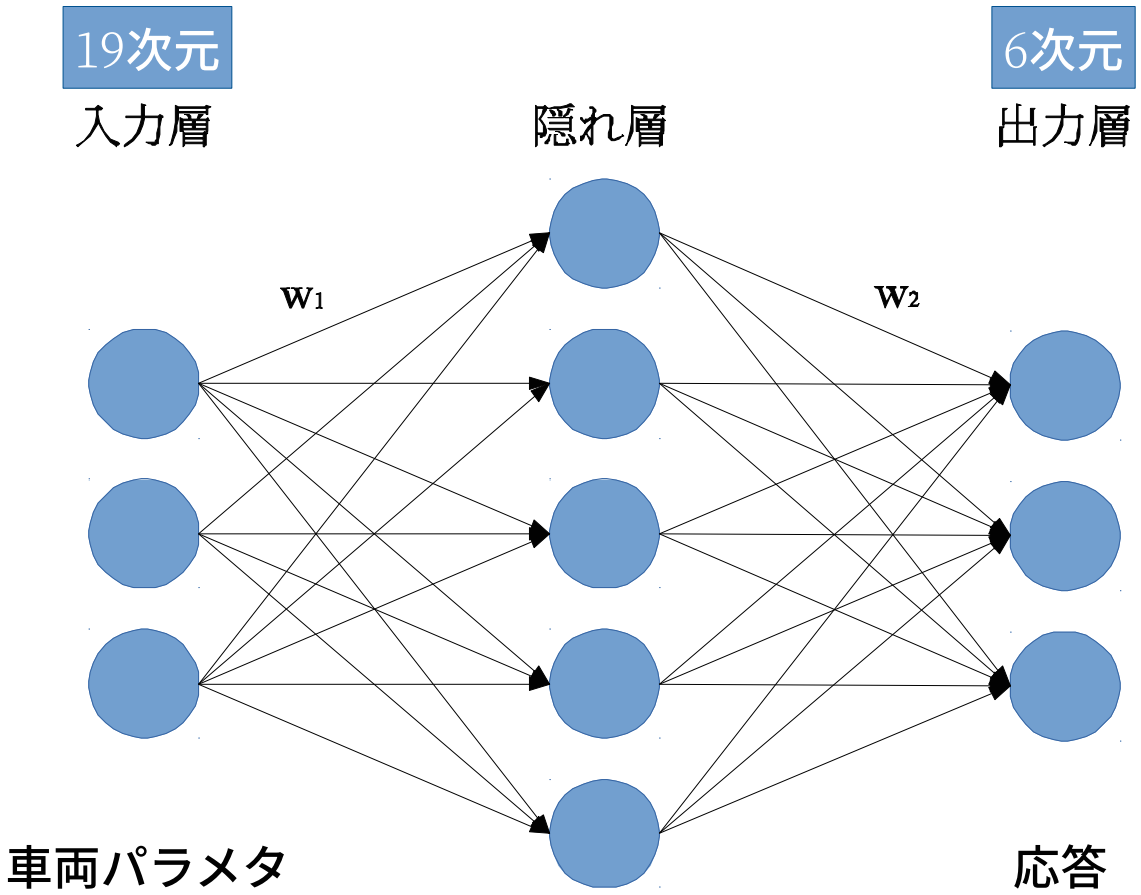

図 3: 隠れ一層ニューラルネットワーク

とすると，単純パーセプトロンは，

$$y = w_0x_0 + w_1x_1 + \cdots + w_nx_n \tag{2.3}$$

$$= \sum_{i=0}^{n} w_ix_i \tag{2.4}$$

ただし，

$$x_0 = 1, w_0 = b \tag{2.5}$$

と表せる．これは線形回帰の式と一致する．近年のモデルでは，非線形関数の重み付き線形和を多層に重ねることで高い性能を得ている．隠れ一層ニューラルネットワークの例を図 3 に示す．

2.3 RNN

Recurrent Neural Network (RNN) はニューラルネットワークの構造の一種で，時系列データを扱うことに長けたモデルである． RNN は内部状態を保持していて図 4 のような構造になっている．内部状態 h を更新していくことで時系列の変遷を捉えることができる．

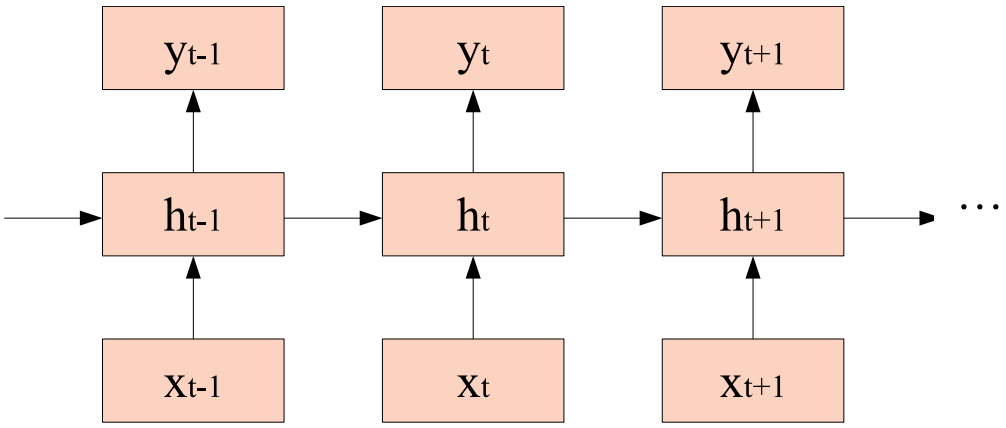

図 4: RNN の概略図

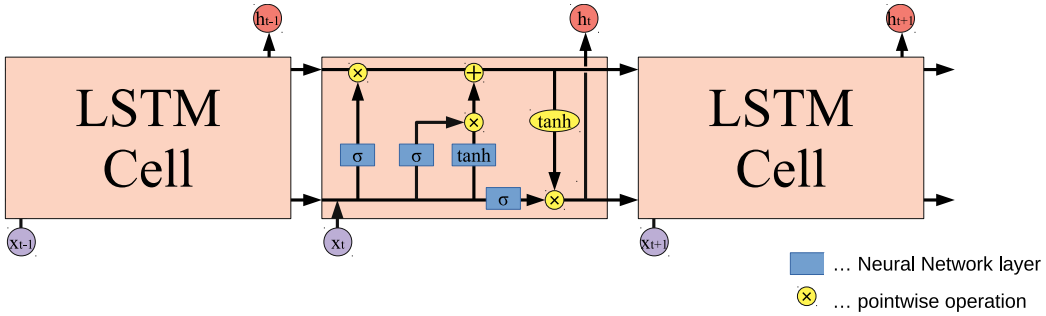

図 5: LSTM の構造の概略図

図 4 の単純な RNN の構造には二点の問題が存在する．一つは計算グラフが一続きとなってしまい，その影響で学習時に勾配消失・爆発が生じてしまうという問題である．もう一つはそのまま内部状態を渡していることで長時間に渡る系列を扱うことができないという問題である．この問題を解決するために提案されたのが Long Term-Short Memory (LSTM) [7] である． LSTM は図 5 のような構造で表される．

LSTM の構造の特徴を述べる．まず通常の RNN と異なる部分として，受け取っている内部状態が 1 つではなく 2 つである点である．図の上側の入力はセル状態と呼ばれ，通常の RNN にはない入力である．セル状態は上部に一直線に通るパスで，このパスによって長期記憶および勾配消失問題の改善ができる．そして，そのパスに対して積を取っている部分によって記憶の忘却が行われる．さらに最後に出力すべきものを選択して出力する，といった構造になっている．この構造によって重要な情報のみを保持していくことで長期に渡る記憶を可能にしている．

2.4 最適化手法

ここではモデルの最適化手法について述べる．

確率的勾配法

一般にニューラルネットワークモデルの学習は勾配法によって最適化を行う．これは，ニューラルネットワークで解く問題は非線形最適化問題であることが多いためである．その中でも確率的勾配法 (Stochastic Gradient Descent; SGD) [8] が用いられることが多い．これは，ニューラルネットワークモデルにデータを入力し，そのときの損失関数の勾配に合わせて重みの更新を行うというものである．確率的勾配法は，

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \eta \Delta \boldsymbol{w} \tag{2.6}$$

と表される．ここで，$\eta > 0$ は学習率，　w_t はステップ t のときのニューラルネットワークモデルの重みである．学習率は学習の進行速度を司るものである． SGD では学習率を人が定める必要がある．学習率を定める際の戦略としては，徐々に局所解を経て大域的最適解を探索するために，学習の最初は大きく，学習が収束し始めたら小さくしていくというものである．

学習率の調整をを人手で行うことはのではなく，機械的に定めてしまうというのが Adam (Adaptive moment estimation) [9] である．これは，勾配に対する慣性を考えることで実現したもので，最近の更新量が大きければ新たな更新量を増やし，小さければ減らす，という考え方に基づいて実現される． Adam を用いると確率的勾配法からハイパーパラメタである学習率を廃することができる．

Adam は以下の式で表せる．

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \cdot g_t \tag{2.7}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) \cdot g_t^2 \tag{2.8}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.9}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.10}$$

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} + \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{2.11}$$

ここで，m は移動指数平均，v は二乗勾配，β_1，β_2 は移動平均の指数減衰率，α はステップサイズ，g は目的関数に対する勾配である．Adam は一次と二次のモーメントを推定するというアイデアから考案された学習アルゴリズムである．学習率がハイパーパラメタから除外されたことと，収束が SGD などの他のアルゴリズムと比較して早いことからよく用いられる．

確率的勾配法を用いる際のデータの入力の仕方について述べる．確率的勾配法では入力されたデータに基づき，そのときの損失関数の勾配によって重みが更新される．データを一つずつ入力して更新することを繰り返す学習方法をオンライン学習という．対して，すべてのデータを同時

に入力，更新していく学習方法をバッチ学習という．しかし，オンライン学習はノイズに対して脆弱で，バッチ学習は局所解に陥りやすいという欠点がある．そこで，その間をとり，あるバッチサイズに基づいて，バッチサイズ分の個数を入力して，その時の勾配で更新することを繰り返し行って学習をする，ミニバッチ学習という学習手法が存在する．これは両者のいい部分を取ることができて，バッチサイズを変更することで両者のバランスを取ることができる．ニューラルネットワークの学習ではミニバッチ学習が用いられていることが多い．

ニュートン法

古典的な最適化手法にニュートン法 [10] がある．ニュートン法は微分可能であれば定義できるため，非線形最適化に対しても用いることができる．二次微分の情報を用いることで，最急降下法と比較して早い収束が期待できる手法である．

$$f(\boldsymbol{\theta}) = 0 \tag{2.12}$$

となる θ を求めることを考える．このとき，ヘッセ行列を $\boldsymbol{H}$ として，

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \boldsymbol{H}^{-1}\nabla f(\boldsymbol{\theta}) \tag{2.13}$$

となるように収束するまで計算を行うと解が得られる，という手法である．ここで得られる解は局所解であり，大域的最適解ではない．そのため，初期値を変えて繰り返し行い，解を定める必要がある．また，ヘッセ行列の逆行列を求める必要があり，パラメタ数が大きくなると計算コストが大きくなる．

レーベンバーグ・マルカート法

ニュートン法と勾配降下法を結合して，両者のトレードオフで最適化を行う手法として，レーベンバーグ・マルカート法 [11] がある．この手法は，局所解に陥りやすいニュートン法と収束が比較的遅い勾配降下法を収束度合いで制御していくことで効率的に最適化を行う手法である．

性能関数が二乗和，つまり Mean Squared Error (MSE) を用いるとき，ヘッセ行列 H は，

$$\boldsymbol{H} = \boldsymbol{J}^{\top}\boldsymbol{J} \tag{2.14}$$

と近似できる．ここで $\boldsymbol{J}$ はパラメタに対するネットワーク誤差のヤコビ行列である．また．勾配 $\boldsymbol{g}$ は

$$\boldsymbol{g} = \boldsymbol{J}\boldsymbol{e} \tag{2.15}$$

と計算できる．ここで．$\boldsymbol{e}$ はネットワーク誤差のベクトルである．ヤコビ行列は逆伝播法で計算できる [12]．これらの計算は式 (2.13) と比較して計算量が少ない．

レーベンバーグ・マルカート法では，ニュートン法と似て，

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - [\boldsymbol{J}^\top \boldsymbol{J} + \mu \boldsymbol{I}]^{-1} \boldsymbol{J}^\top \boldsymbol{e} \tag{2.16}$$

といったように更新が行われる．μ の大小で学習が制御され，$\mu = 0$ のとき，近似ヘッセ行列を用いるニュートン法，μ が大きいとき，学習率が小さい勾配降下法となる．そのため，学習が進むに連れて μ は小さくしていかなければならない．

2.5 正則化

機械学習分野での正則化は過学習を防止するために用いられる手法のことを指す．主要な正則化について述べる．機械学習ではパラメタ数が多いと図 6 のように過適合が発生することがある．パラメタの大きさなどに罰則をかけることで汎化性能を上げ，過学習を防ぐことを行う．それが正則化である．

ニューラルネットワークに用いられる正則化の手法の一つとして，ドロップアウト [13] がある．ランダムでニューラルネットワークのユニットを削除することで，それぞれのユニットが分割して学習するようになり，全体の汎化性能が向上するという手法である．

また， L2 正則化もニューラルネットワークの正則化によく利用される． L2 正則化は罰則項として重みの L2 ノルムを目的関数に加えるという手法である．この L2 正則化項は，極端に大きな重みが存在する場合には過適合している可能性が高いという前提で導入される．なので，データに適合はしにくくなるが，汎化性能が上昇するという効果が見込める．

2.6 スキップコネクション

スキップコネクションとは，層を飛ばした結合のことである．スキップコネクションは主にふたつの理由で用いられる．一つはニューラルネットワークの最適化の際に勾配を伝えるためである．He ら [14] はスキップコネクションを用いることで勾配消失問題を解決し，層を深くすることで高い性能を持つモデルを構築した．もう一つは層の間で情報を共有することで解を簡単化したりするために用いられる．Huang ら [15] は層の間で密にスキップコネクションを加えることで情報を共有させ，性能を向上させている．また，Li ら [16] は損失関数を可視化することによって，解が簡単化されていることを確認している．

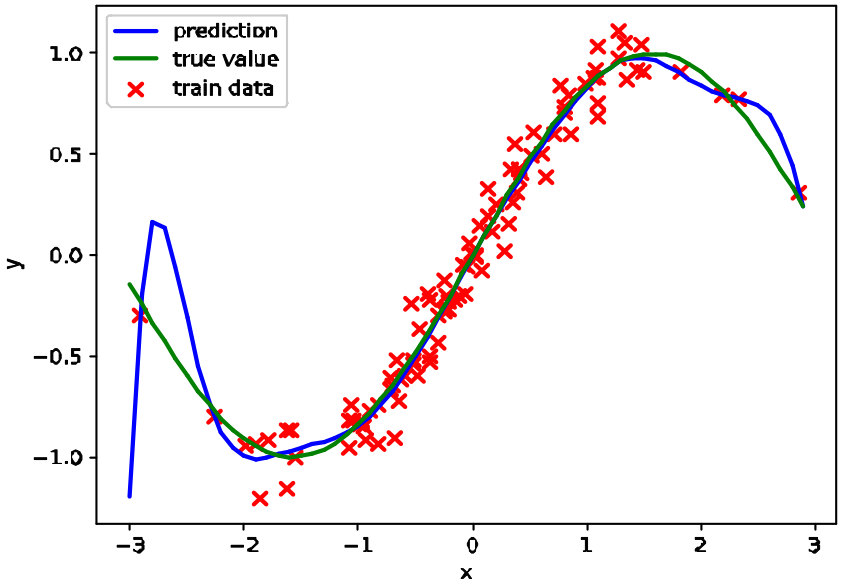

図 6: 正則化による過学習の抑制

2.7 マルチタスク学習

予測の精度を向上させるためにマルチタスク学習 [17, 18] という方法が用いられることがある．これは複数のタスクをモデルに学習させ，共通情報の共有をさせることで精度を向上させるという手法である．Xu らの手法 [17] では，物体検出と分類を同時に行っている．これは特徴量マップを一致させることで実現していて，その特徴量が物体検出と分類を解くことができる特徴量になっている．その共有した特徴量によって情報を共有し，解を得るための情報を増やすことができるので精度を向上させることができる．

2.8 先行研究

本研究は同一のタスクに対して先行研究が行われている．このモデルを本研究のベースラインとする．ここで用いられているモデルは直接，車両パラメタから応答を求めるというものである．用いているモデルは図 3 のような隠れ一層ニューラルネットワークである．この隠れ一層ニューラルネットワークを 2.4 節のレーベンバーグ・マルカート法で最適化する．入力は車両パラメタの 19 次元，出力は応答の 6 次元であるが（データの詳細は 3.1 節で説明する），それぞれの応答は独立に扱うものとしてそれぞれのパラメタに対して別のモデルを用いることとする．つまりモデルは

6 つ用意され，19 次元の入力に対して一つの応答を求めるモデルが構築されるということである．このモデルは隠れ層のユニット数に対してチューニングされる．損失関数は，MSE を用いている．

第3章　提案手法

本研究では機械学習を用いた車両走行シミュレーションに対して系列モデルを導入する．従来手法では車両パラメタから直接に応答を求めていた．精度を向上させるために系列モデルを導入し，中間で計算されている三次元位置やロール角などの系列データを計算することで精度向上を図る．3.2 節ではこの系列モデルを用いたモデルについて述べる．加えてその提案モデルに対する提案として，3.3 節ではモデルに対する損失関数，3.4 節ではモデルに導入したスキップコネクション，3.5 節ではこのモデルを学習するための学習方法であるスケジュールドサンプリングについて述べる．

3.1　タスク設定

本研究では車両走行シミュレーションの中でも，自動車の動的転倒傾向の調査を目的とする試験である NCAPFishhook 試験を対象とする．この試験は，世界的に標準的に用いられている試験であり，自動車メーカーはこの試験をクリアしなければ製品化することはできない．この試験に対してタイヤの摩擦係数やサスペンションのばね係数などのパラメタを最適化するため，パラメタを変化させたときに NCAPFishhook 試験における応答を得ることを本研究の対象タスクとする．

本研究で行うモデル化の対象について述べる．手順 2 の釣り針型の経路での走行過程に比べて，手順 1 入力ステアリング角の決定過程では，あまり計算量が多くないため十分シミュレータで計算することができる．そこで本研究のモデル化の対象は釣り針型の走行手順部分のみとする．精度よく応答を得るために手順 1 はシミュレータを用いて事前計算を行うこととする．

図 7 は本研究のデータの生成方法の概要である．入力は車両パラメタで，事前計算によって入力ステアリング角を規定する．そして車両パラメタとステアリング角をシミュレータに入力すると，出力として中間計算パラメタの時系列データが得られる．中間計算パラメタのうちの 6 パラメタに対して最大値や最小値を取ることで最終的な 6 次元の応答を得ることができる．

このようなシミュレーションに対して機械学習を用いる手法では，シミュレータを模するのではなく直接出力を得る手法が取られることがあった．[2, 3] そこで本研究では，シミュレータを模して，中間計算パラメタの時系列データを推定してから得るべき応答を推定するという過程でモデルを構成する．シミュレータのように中間計算パラメタを経て応答を予測することで，応答の推定に必要な情報を増やすことができ，精度の向上が期待できる．

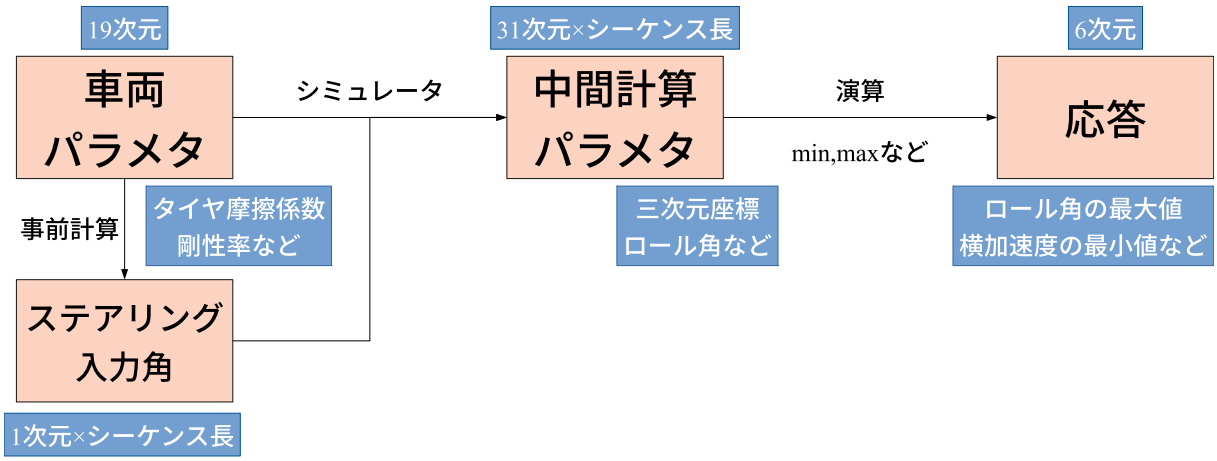

図 7: データの生成方法の概要

3.2 系列モデルを用いたモデル

提案モデルの概略を図 8 に示す．本提案モデルは大きく分けて二段階に分かれて計算が行われている．シミュレータを模しているシミュレータ部とその系列データから最終的な応答を抽出するための演算部である．まず入力からシミュレータ部で中間計算パラメタである系列データを計算する．そしてその系列データから応答を演算部で抽出するという流れで計算が行われる．

入力

まず入力部について説明する．各ステップにおける入力は車両パラメタとそのステップにおけるステアリング入力角と直前の 2 ステップの中間計算パラメタである．車両パラメタは一つの系列の中では同一の値であり，全てのステップで同一の値を入力する．車両パラメタとステアリング入力角を結合したものを $\boldsymbol{x}$，タイムステップを t とし，t における $\boldsymbol{x}$ を $\boldsymbol{x}_t$ と表すこととする．車両パラメタを $\boldsymbol{p} = (p_1, p_2, \cdots p_N)$，ステアリング入力角を $s = \{s_1, s_2, \cdots, s_T\}$ とすると $\boldsymbol{x}_t$ は，

$$\boldsymbol{x}_t = (s_t, p_1, p_2, \cdots, p_N)^\top \tag{3.1}$$

と表せる．ただし，N は車両パラメタの次元数，T は系列長である．

シミュレータ部

入力に対してシミュレータを模して順次計算し，中間計算パラメタを計算しているのがシミュレータ部である．このシミュレータ部は結合層と LSTM によって構成される．結合層は入力を特徴付ける特徴量を得るための層で，入力から特徴量を抽出する．LSTM はシミュレータの順次計算を模倣するための系列モデルである．結合層のニューラルネットワークを NN($\boldsymbol{x}$)， LSTM を

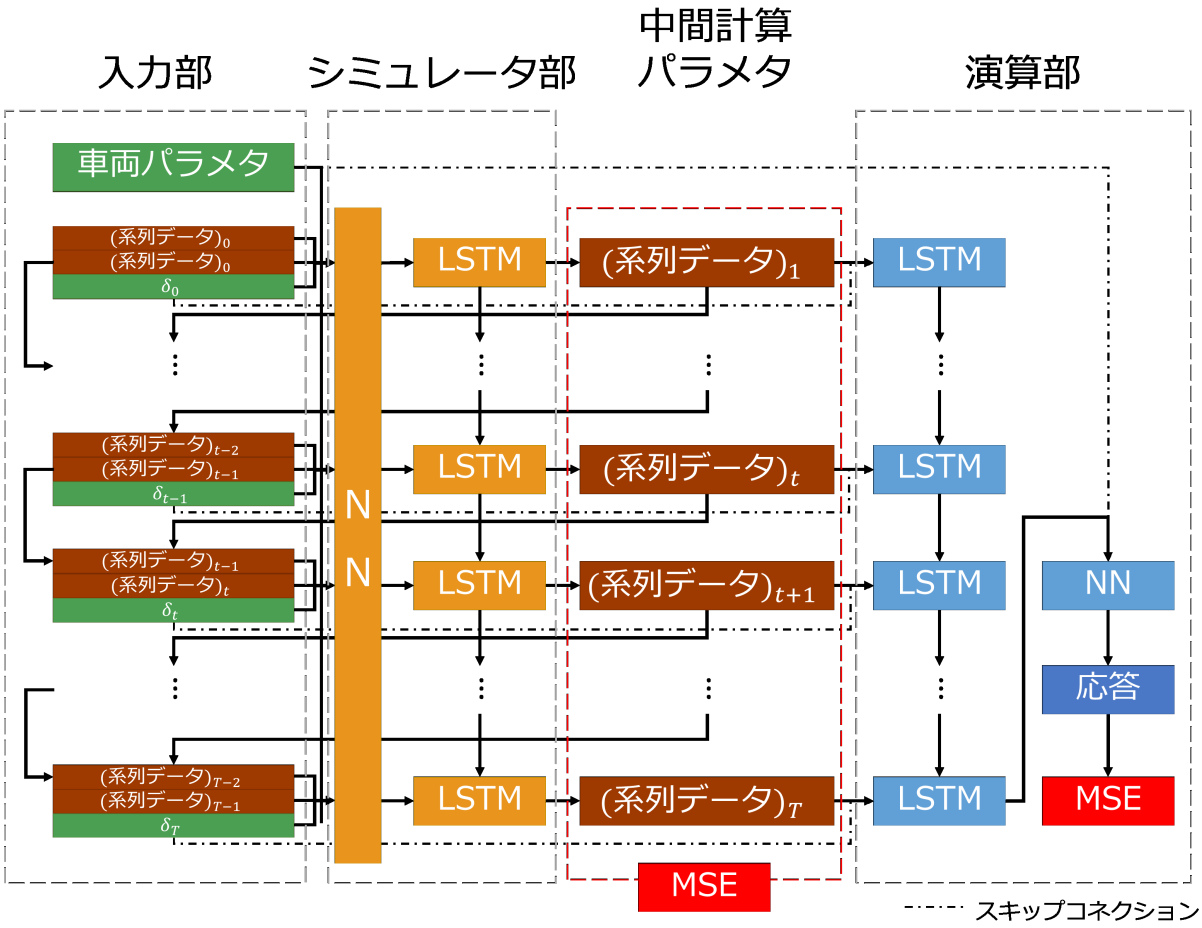
入力部
シミュレータ部
中間計算
パラメタ
演算部
車両パラメタ
(系列データ)$_0$
(系列データ)$_0$
δ_0
(系列データ)$_{t-2}$
(系列データ)$_{t-1}$
δ_{t-1}
(系列データ)$_{t-1}$
(系列データ)$_t$
δ_t
(系列データ)$_{T-2}$
(系列データ)$_{T-1}$
δ_T
NN
LSTM
LSTM
LSTM
LSTM
(系列データ)$_1$
(系列データ)$_t$
(系列データ)$_{t+1}$
(系列データ)$_T$
MSE
LSTM
LSTM
LSTM
LSTM
NN
応答
MSE
スキップコネクション

図 8: 提案モデル

LSTM($\boldsymbol{x}$) と置く．ただし，結合層は隠れ層が N 層で，重み行列を W，非線形関数の活性化関数を $f(x)$ とすると NN($\boldsymbol{x}$) は，

$$\mathrm{NN}(\boldsymbol{x}) = \mathrm{f_N}(\mathrm{W_N f_{N+1}}(\mathrm{W_{N-1}} \cdots \mathrm{f_1}(\mathrm{W_1 x}) \cdots)) \tag{3.2}$$

と表せる．このシミュレータ部のステップ毎の出力 $\hat{\boldsymbol{z}}_t$ は，

$$\hat{\boldsymbol{z}}_t = \mathrm{LSTM}(\mathrm{NN}(\boldsymbol{x}_{\mathrm{t-1}}, \hat{\boldsymbol{z}}_{\mathrm{t-1}}, \hat{\boldsymbol{z}}_{\mathrm{t-2}})) \tag{3.3}$$

となる．このとき，$\hat{\boldsymbol{z}}$ は中間計算パラメタの推定値である．また，順次計算をするために必要のないふたつ前の中間計算パラメタも入力しているのは，推定パラメタの微分をモデルが保持するためである．しかし，初期値は一つであるからふたつ前の中間計算パラメタは存在しない．時刻ゼロの時点の中間計算パラメタ $\hat{\boldsymbol{z}}_0$ は事前の計算によって得ることができる．そこで，シミュレーションの初期は緩やかな動作である，という仮定から

$$\hat{\boldsymbol{z}}_1 = \mathrm{LSTM}(\mathrm{NN}(\boldsymbol{x}_0, \hat{\boldsymbol{z}}_0, \hat{\boldsymbol{z}}_0)) \tag{3.4}$$

とすることでこの問題を解決する．

演算部

提案モデルでは，得られた中間計算パラメタの系列データに対して応答を end-to-end に抽出することを行っている．順次計算によって得られた中間計算パラメタの系列データを LSTM に入力し，その系列から応答を得るために必要な特徴量を抽出する．その特徴量から結合層で応答を計算する．演算部の出力を $\hat{\boldsymbol{y}}$ とすると，

$$\hat{\boldsymbol{y}} = \mathrm{NN}(\mathrm{LSTM}(\hat{\boldsymbol{z}})) \tag{3.5}$$

と表せる．ただし，$\hat{\boldsymbol{y}}$ は応答の推定値である．

3.3 ハイブリッド損失

損失関数は中間計算パラメタと応答を同時に最適化できるようにハイブリッドな損失を用いる．このモデルでは中間計算パラメタの系列データと応答の両者を計算している．そこで用いている損失関数 L が以下の (3.6) 式である．

$$L = \alpha \mathrm{MSE}(\boldsymbol{z}, \hat{\boldsymbol{z}}) + \beta \mathrm{MSE}(\boldsymbol{y}, \hat{\boldsymbol{y}})) \tag{3.6}$$

ここで，$\boldsymbol{y}$ は応答の真値，$\boldsymbol{z}$ は中間計算パラメタの真値，α と β は係数で任意の定数である．また，$\mathrm{MSE(y, \hat{y})} = \frac{1}{\mathrm{N}}(\mathrm{y} - \hat{\mathrm{y}})^2$ である．

3.4 スキップコネクション

提案モデルではスキップコネクションを加えることで精度向上を図っている．本提案モデルのスキップコネクションは，層を通すことで欠落してしまった情報を付加し直して解を簡単化することを目的に加えられている．提案モデルは 3.2 節で述べたとおり，中間計算パラメタを計算するシミュレータ部と応答を抽出する演算部の二段階で構成されている．そのため，シミュレータ部は系列データを得るために最適化され，演算部に到達するときには入力の情報が欠落してしまっていることが考えられる．そこで，車両パラメタとステアリング入力角を演算部に渡すためにスキップコネクションを用いる．

図 8 の一点鎖線部がスキップコネクションである．ステアリング入力角は演算部の LSTM に入力される系列データに結合するようにスキップコネクションを用いている．車両パラメタは抽出された特徴量から応答を計算する部分で特徴量に結合するようにしている．このときの演算部の式 (3.5) を書き直すと，

$$\hat{\boldsymbol{y}} = \mathrm{NN}(\mathrm{LSTM}(\hat{\boldsymbol{z}}, \mathrm{s}), \boldsymbol{p}) \tag{3.7}$$

となる．

3.5 スケジュールドサンプリング

本提案モデルではシミュレータの順次計算を模倣するために，一ステップ前の中間計算パラメタの予測値を入力として扱う．しかし，この手法には問題があると考えられる．それは学習時に，誤差が大きく推定できていいない予測値を入力として扱うことで，モデルが誤差を取り除くために学習することとなってしまうという問題である．

そこで，学習初期は予測値を入力とするのではなく，正解の値を入力として扱う．正解を入力として扱うことで，ステップ毎には誤差が少ない解にモデルは最適化される．ただし，このように正解を入力とすると，そのステップのみで誤差が少ない解になってしまい，全体的に誤差が少ない解となることはない．

全体で誤差が少ない解にするためには予測値を入力として扱い，全体的な損失を最小化する必要がある．このトレードオフを解決するために，本研究では学習初期には正解を入力とし，学習中期には正解と予測値をランダムで入力として扱い，学習終盤には予測値のみを入力とする，スケジュールドサンプリング [19] を行うことで全体的に誤差が少ない解へと徐々に導く．

正解を入力として扱った場合のシミュレータ部の (3.3) 式を書き直すと，

$$\boldsymbol{z}_t = \mathrm{LSTM}(\mathrm{NN}(\boldsymbol{x}_{\mathrm{t}-1}, \boldsymbol{z}_{\mathrm{t}-1})) \tag{3.8}$$

となる．このスケジュールドサンプリングは，学習時にはシミュレータ部を (3.8) 式から (3.3) 式へと入力を推定値から真値へ変化させて，テスト時には式 3.3 のように推定値のみを入力として扱う．

第4章　モデルの評価

本章では 3 章の提案手法のそれぞれの効果について検証する．4.1 節では実験の設定を，4.2 節では検証の結果を述べる．

4.1　共通の実験設定

実験環境

提案手法の実験のプログラムは，プログラム言語 Python version 3.6.6 を用いて実装した．用いた機械学習フレームワークは PyTorch version 1.0 である．グラフィックのドライバは NVIDIA ディスプレイドライバ version 410.93，用いた CUDA (Compute Unified Device Architecture) は CUDA version 10.0 である．用いた計算機のスペックを表 1 に示す．

PyTorch は機械学習のフレームワークであり，テンソル操作のためのライブラリである．特徴として，テンソル操作を行うと計算グラフを保持し，ニューラルネットワークの学習時には誤差を逆伝搬させるための勾配を自動で微分して計算することができる．この特性から， Define by Run 型のフレームワークと呼ばれ，ネットワークを構築するに当たって，順伝搬の計算のソースを記述すると逆伝搬の計算を記述する必要がない．本研究ではネットワーク構造を変更して実験を行うことが多かったため，このフレームワークを用いて実装期間の短縮に努めた．

2.8 節の従来手法は MATLAB を用いて実装されている．実行環境のハードウエア構成は提案手法と同一で表 1 のものである．

データセットは図 7 のように，入力と中間計算される系列データと出力となる応答からなる．入力は 19 次元の車両パラメタと系列データのステアリング入力角，中間計算パラメタの系列データは 31 次元，その中間計算パラメタの時系列データから最大値や最小値などの演算をして得られる最終的な応答が 6 次元である．前処理として，すべてのデータは平均 0，分散 1 に正規化した．データセットの構成は，表 2 の通りである．

表 1: ハードウエア構成

種別	ハードウエア
中央演算装置	Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz
Graphic Processing Unit; GPU	GeForce GTX TITAN X (Pascal)
主記憶装置	64GB

評価指標は応答の平均平方誤差 (Root Mean Square Error; RMSE) を用いた．RMSE は，

$$RMSE(y, \hat{y}) \quad = \quad \sqrt{\frac{1}{D}\sum_{i=1}^{D}(\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i)^2} \tag{4.1}$$

と表せる．ただし，D はデータの数である．それぞれの応答パラメタにおける誤差 $e_i(i = 1, 2, \cdots, 6)$ は，

$$e_i = RMSE(y_i, \hat{y}_i) \tag{4.2}$$

となる．モデルの性能をわかりやすく見るために，応答パラメタごとの RMSE の平均 $\bar{e}$ を定義する．

$$\bar{e} = \frac{1}{6}\sum_{i=1}^{6} e_i \tag{4.3}$$

この値が小さければ，モデルの推定精度が高いことを示し，検証したモデルが有効であることが確認できる．加えてパラメタ毎の推定精度のばらつきを見るため，パラメタ毎の誤差の分散 σ_e^2 を定義する．

$$\sigma_e^2 = \sqrt{\frac{1}{6}\sum_{i=1}^{6}(e_i - \bar{e})^2} \tag{4.4}$$

実験手順は，構築したモデルの重みを Pytorch の初期乱数で初期化し，学習した後にテストを行う．最適化手法は Adam を用いた．スコアは，1 エポック毎に求め，最も小さい $\bar{e}$ となったときのスコアをその試行に対するスコアとし，3 回試行した平均でスコアを定める．σ_e^2 については，推定誤差の分散なので，平均を取ったあとの e_i を用いて求めている．

ネットワークの構成は表 3 に示した．シミュレータ部は入力から結合層によって特徴量となり，LSTM に入力，中間計算パラメタの次元に落とすために結合層を入れて中間計算パラメタが得られる．中間計算パラメタを得るための結合層では，正則化としてドロップアウトを加えている．演算部は中間計算パラメタを入力として，結合層で次元を増やし， LSTM で可変長の系列を特徴量に落としこむ．そして，ドロップアウトの正則化を加えた結合層で最終的な出力である応答を得る．

表 2: データセットの構成

訓練	評価	テスト	合計
600	76	76	752

4.2 実験結果

本研究の提案手法の評価の結果を述べる．4.2.1 節では，従来手法と提案手法を比較して，本研究の有効性を確認する．5 節では導入した中間計算パラメータの効果について検証した．6 節ではスキップコネクションを導入した効果，4.2.4 節ではスケジュールドサンプリングの効果について述べる． 4.2.5 節では系列モデルを導入したことによる効果を検証し，4.2.6 節では (3.4) 式のように行った，シミュレーション初期は運動が穏やかであるという仮定について検証する．

4.2.1 従来手法と提案手法の比較

本提案手法による精度の向上の確認のため，2.8 節の直接に車両パラメタから応答を求めるモデルを用いた従来手法と提案手法を比較する．結果は表 4 に示した．表の数値が太字になっている方が誤差を少なく，精度良く推定できているということである．表 4 を見ると一つの応答パラメタを除いて従来手法に比べて精度が向上していることがわかる．

加えて提案手法はパラメタごとに誤差の大きさが異なることがわかる．σ_e^2 を見ると，提案手法のパラメタの推定精度の分散が大きいことが読み取れる．これはパラメタによって推定しやすいものとしにくいものがある，つまり推定難度が異なるということが考えられる．

ひとつ目のパラメタは誤差が従来手法と比べて増加してしまった，つまり精度が下がってしまった．この原因として考えられるのは，データ量の不足が考えられる．従来手法のモデルは隠れ一層ニューラルネットワークであった．提案手法はパラメタ数が多く，精度が下がってしまったことが考えられる．

表 3: ネットワークの構成

	ID	名前	次元
シミュレータ部	1	結合層 + ReLU	512
	2	LSTM + ReLU	512
	3	結合層 + ドロップアウト (0.1)	512
		出力 (中間計算パラメタ)	31
演算部	4	結合層 + ReLU	512
	5	LSTM + ReLU	512
	6	結合層 + ドロップアウト (0.1)	512
		出力 (応答)	6

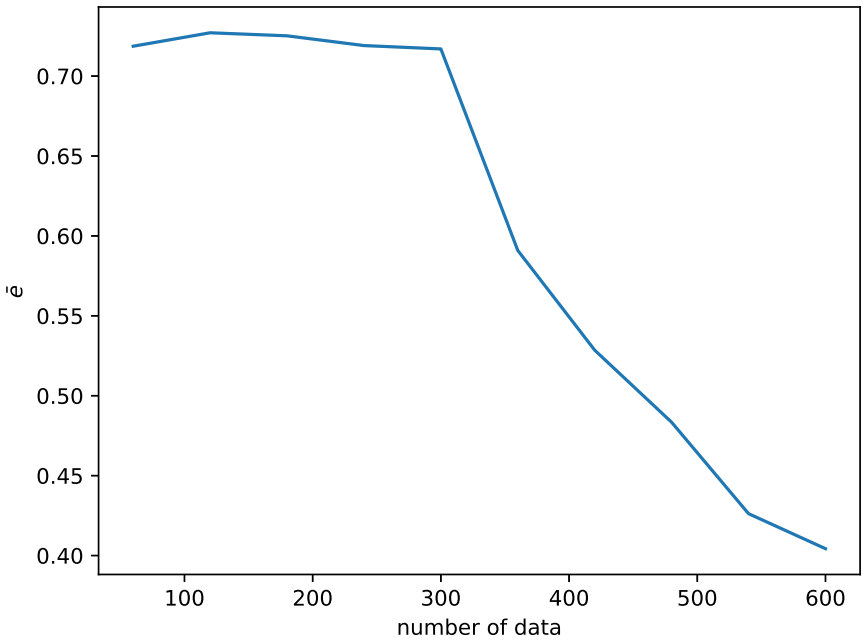

図 9: データ数と推定誤差の平均 $\bar{e}$ の関係

図 9 はデータ数と推定誤差の平均の関係で，図 10, 11, 11, 12, 13, 14,15 はそれぞれ推定パラメタに対するデータ数と推定誤差の関係である．これらの関係を見ると，すべてのデータを使っているが，まだ精度が飽和していない状態にあることがわかる．このことからこのモデルを訓練するにはデータが足りていないことがわかる．これを解決するためには，データを増やすか，モデルを小さくして少ないデータでも訓練できるモデルにする必要がある．ひとつ目のパラメタについては，図 10 を見ると，飽和する兆しは見えるが，まだデータが足りているかは未定であるのでデータを増やして検証する必要がある．

4.2.2 ハイブリッド損失の効果検証

4.2.1 節では従来手法より提案手法の精度が向上したことを確認した．この節では中間計算パラメタの情報を付加することによる効果を確認する．従来手法と提案手法ではモデルのパラメタ数

表 4: 従来手法と提案手法の推定誤差の比較

モデル	e_1	e_2	e_3	e_4	e_5	e_6	$\bar{e}$	σ_e^2
従来手法	**0.6165**	0.4915	0.4605	0.5555	0.3492	0.5608	0.5057	0.0089
提案手法	1.364	**0.0980**	**0.1073**	**0.3055**	**0.1413**	**0.4092**	**0.4042**	*0.2364*

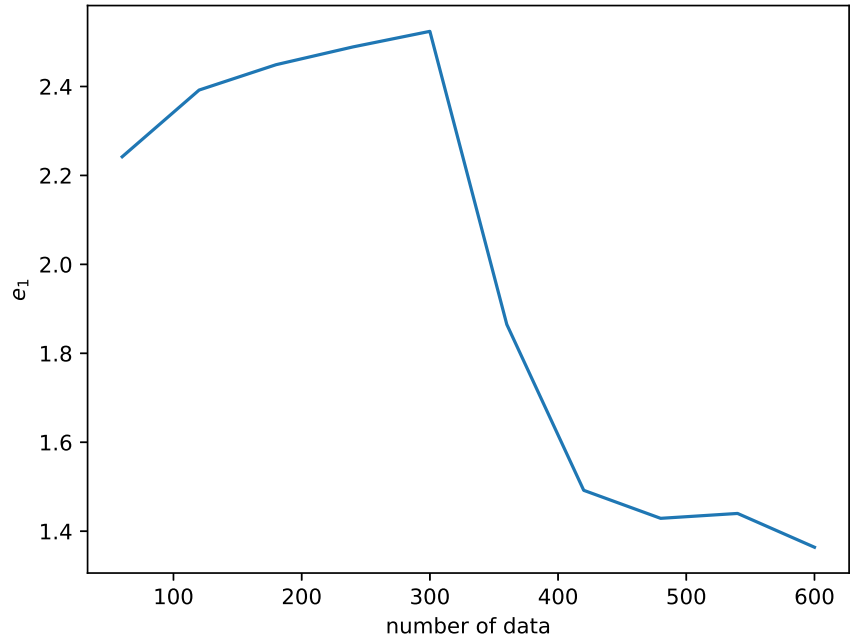

図 10: データ数と推定パラメタ 1 における推定誤差 e_1 の関係

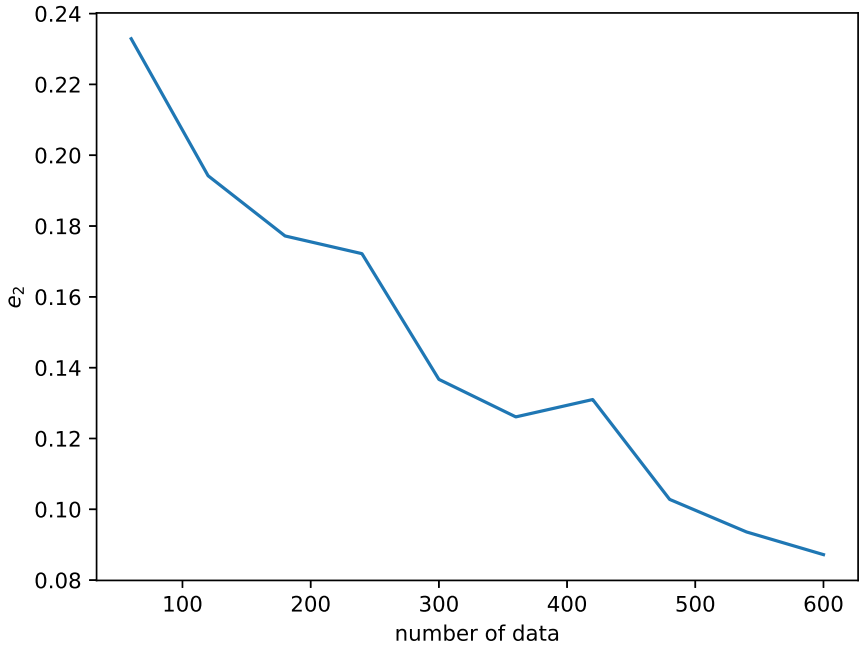

図 11: データ数と推定パラメタ 2 における推定誤差 e_2 の関係

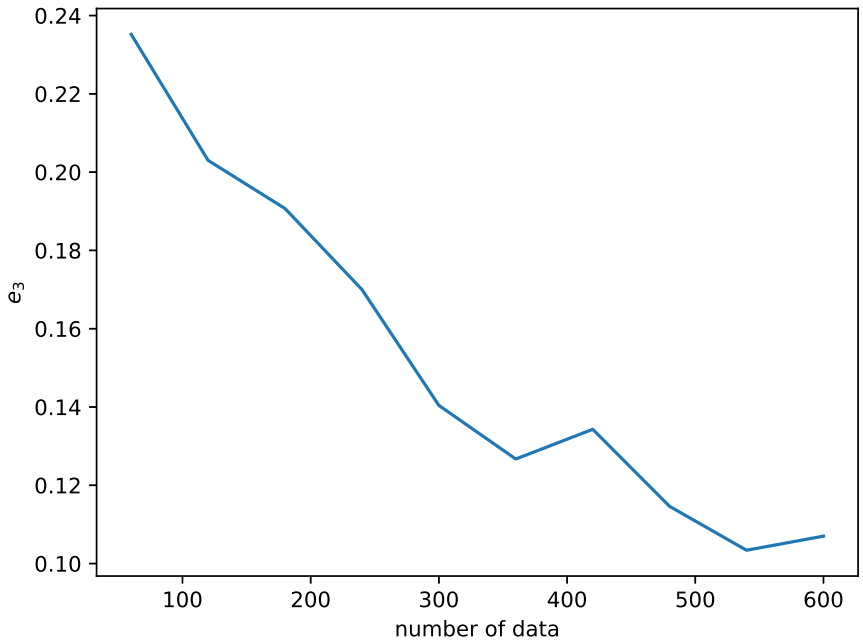

図 12: データ数と推定パラメタ 3 における推定誤差 e_3 の関係

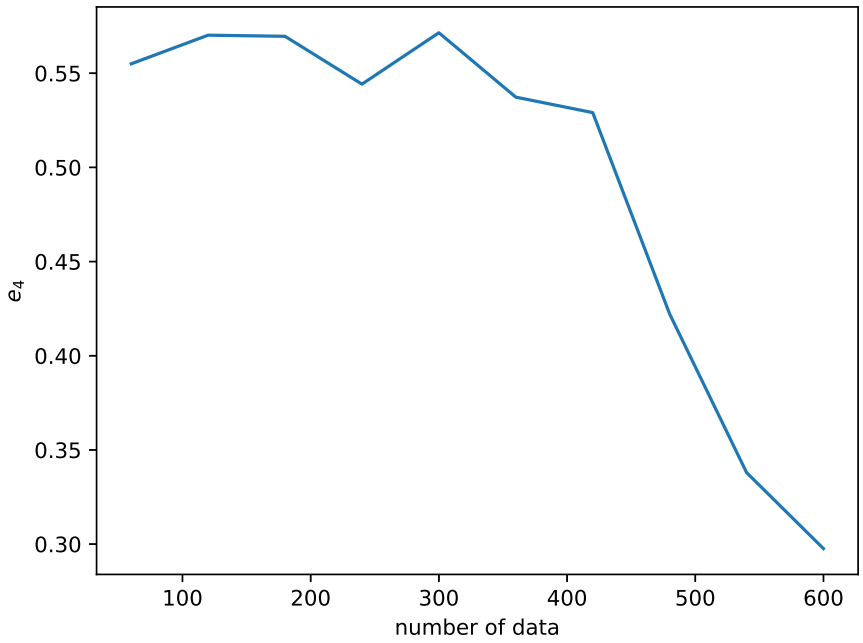

図 13: データ数と推定パラメタ 4 における推定誤差 e_4 の関係

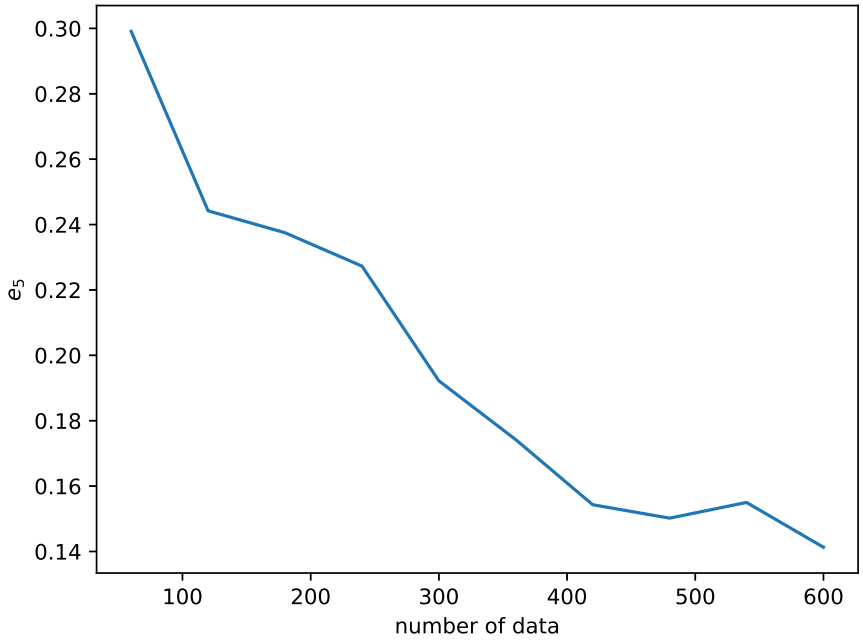

図 14: データ数と推定パラメタ 5 における推定誤差 e_5 の関係

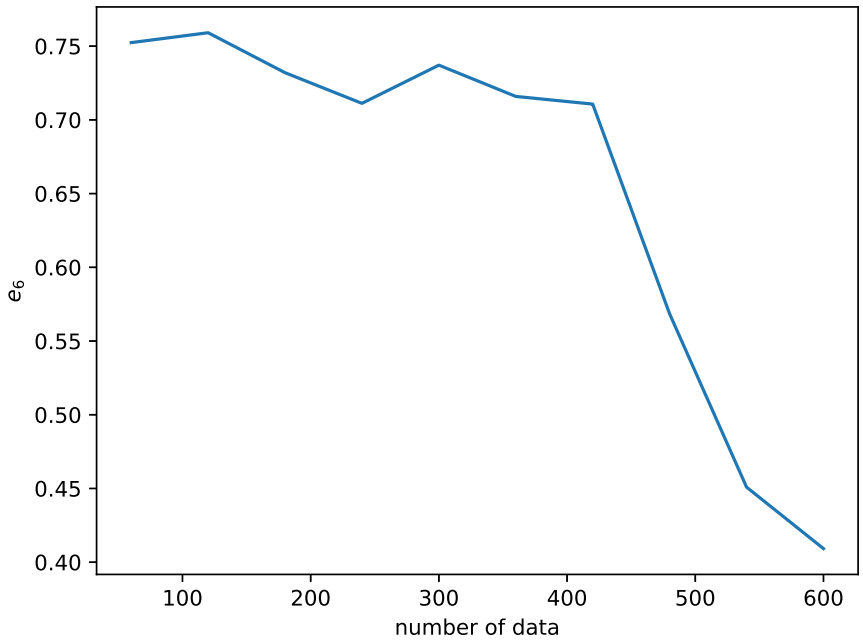

図 15: データ数と推定パラメタ 6 における推定誤差 e_6 の関係

が大きく異なる．その寄与によって精度が向上したのか，中間計算パラメタを付与したことによる精度の向上なのかを検証する必要がある．

ここでは 3.3 節のハイブリッド損失 (3.6) 式を用いた場合と，(3.6) 式において $\alpha = 0$ とした応答のみの損失を用いた場合の比較を行う．この実験によって，中間計算パラメタを計算していることによる効果が確認できる．応答のみの損失は以下の (4.5) 式の損失のようになる．

$$\begin{aligned} L &= \alpha MSE(\boldsymbol{z}, \hat{\boldsymbol{z}}) + \beta MSE(\boldsymbol{y}, \hat{\boldsymbol{y}}))|_{\alpha=0} \\ &= \beta MSE(\boldsymbol{y}, \hat{\boldsymbol{y}}) \end{aligned} \tag{4.5}$$

この実験の結果が表 5 である．結果を見ると，ハイブリッド損失を用いた場合のほうが誤差が大きくなってしまっている推定パラメタもあるが，全体ではハイブリッド損失を用いている場合のほうが誤差が小さくなっている．ハイブリッド損失を用いていないものと従来手法と比較すると精度が向上していることがわかる．よって，ハイブリッド損失を用いることによって中間計算パラメタの情報を与えることで精度の向上はするが，パラメタ数による寄与のほうが大きいということがわかった．従来手法はパラメタ数が足りておらず，表現力が足りていない．本研究の提案手法ではパラメタ数が増えていることから大きく精度が向上していると言える．

4.2.3 スキップコネクションの効果検証

提案手法は 3.4 節のスキップコネクションを用いている．これは，シミュレータ部を通すことで最初の入力の情報が落ちている状態で演算部に入力されることを防ぐためであった．このスキップコネクションによって精度が向上したのか，もしくは低下したのかを検証する．(3.7) 式のようにステアリング角と車両パラメタをそれぞれ応答部の LSTM と結合層に入力するモデルと，(3.5) 式のように入力しないモデルで比較する．

その結果が表 6 である．この結果を見ると，一つのパラメタを除いた五つのパラメタでスキップコネクションが有るモデルの方が精度が高い，つまり推定誤差が小さい．全体的な誤差の大き

表 5: ハイブリッド損失 (3.6) 式を用いた場合と応答のみの損失 (4.5) 式を用いた場合の比較

モデル	e_1	e_2	e_3	e_4	e_5	e_6	$\bar{e}$	σ_e^2
従来手法	0.6165	0.4915	0.4605	0.5555	0.3492	0.5608	0.5057	0.0089
$\alpha = 0$	1.442	0.1034	0.1133	**0.2946**	0.1507	**0.3957**	0.4166	0.2655
ハイブリッド損失	**1.364**	**0.0980**	**0.1073**	0.3055	**0.1413**	0.4092	**0.4042**	0.2364

さを比較してもスキップコネクションが有るモデルの方が推定精度が高いことがわかる．このことから，スキップコネクションを加えることで，シミュレータ部を通したことによる情報の欠落を補い，精度を向上することができたといえる．

4.2.4 スケジュールドサンプリングの効果検証

3.5 節のスケジュールドサンプリングの効果を検証する．ここで検証するのはシミュレータ部の入力の一つ前の状態の入力を推定値と真値をどのように入力するかということである．検証するのは三種類の入力の仕方である．

(1) 常に推定値を入力

常にシミュレータ部の入力に推定値を用いる．このとき，(3.3) 式にしたがって訓練及びテストが行われる．

(2) 常に真値を入力

常にシミュレータ部の入力に真値を用いる．このとき，訓練時は (3.8) 式に従い，テスト時は (3.3) 式に従う．

(3) 徐々に真値から推定値へと入力を変化

シミュレータ部の入力を真値から推定値へと徐々に変化させる．変数 k がベルヌーイ分布，

$$f(k;q) = q^k(1-q)^1 - k \tag{4.6}$$

に従うとすると，入力の変化のさせ方は，

$$\boldsymbol{z}_t = \begin{cases} \mathrm{LSTM}(\mathrm{NN}(\boldsymbol{x}_{\mathrm{t}-1}, \boldsymbol{z}_{\mathrm{t}-1})), & if\ k = 1 \\ \mathrm{LSTM}(\mathrm{NN}(\boldsymbol{x}_{\mathrm{t}-1}, \hat{\boldsymbol{z}}_{\mathrm{t}-1})), & otherwize \end{cases} \tag{4.7}$$

表 6: スキップコネクションの有無による推定誤差の比較

モデル	e_1	e_2	e_3	e_4	e_5	e_6	$\bar{e}$	σ_e^2
従来手法	0.6165	0.4915	0.4605	0.5555	0.3492	0.5608	0.5057	0.0089
スキップコネクション無	1.701	0.1571	0.1292	0.4243	0.2042	0.5439	0.5266	0.3577
スキップコネクション有	**1.364**	**0.0980**	**0.1073**	**0.3055**	**0.1413**	**0.4092**	**0.4042**	0.2364

と表せる．ただし，

$$q = \begin{cases} 1, & \frac{epoch}{NUM\_EPOCHS} < \frac{1}{3} \\ 0.5, & \frac{1}{3} \leq \frac{epoch}{NUM\_EPOCHS} < \frac{2}{3} \\ 0, & \frac{2}{3} \geq \frac{epoch}{NUM\_EPOCHS} \end{cases} \tag{4.8}$$

とする．このとき，NUM_EPOCHS は全体のエポック数，$epoch$ は現在のエポック数である．これは，学習時のステップを三段階に分け，学習初期は真値，学習中期は真値と推定値を 1 : 1 で入力，学習終盤は推定値のみを入力する．

この三つについて実験を行った結果が 7 である．結果の平均を見ると，最も精度よく推定できているのは (3) であるといえる．しかし，それぞれの推定パラメタを見ると (2) が最も多くのパラメタで良い精度を出していることがわかる．そこで σ_e^2 を見ると分散が最も小さいのは (3) であった．つまり，それぞれの推定パラメタに対して全体として精度良く推定できるモデルになっていることがわかる．このことからスケジュールドサンプリングを行うことで安定した学習ができていると言える．

4.2.5 シミュレータ部での LSTM の有効性検証

このタスクに対する LSTM によるシミュレータ部の系列学習の有効性を確認する．図 16 のように，シミュレータ部で系列を考慮せずに一つ前の状態から次の状態を求める，マルコフ性を仮定したモデルを用いてシミュレータ部における効果の検証を行う．このときのモデルが図 16 のモデルである．モデルの詳細は表 8 に示した．図 8 の提案モデルと比較すると，シミュレータ部の LSTM の有無が異なっている．

このときの実験結果が表 9 である．結果を見ると，僅かな差で LSTM を用いたモデルの方が誤差が大きくなってしまっている推定パラメタがあるが，全体を見ると LSTM を用いたモデルの方が誤差が小さいことがわかる．このことから本タスクに対しては，系列全体を考慮してモデル化

表 7: スケジュールドサンプリングの学習時の入力による誤差の比較

モデル	e_1	e_2	e_3	e_4	e_5	e_6	$\bar{e}$	σ_e^2
従来手法	0.6165	0.4915	0.4605	0.5555	0.3492	0.5608	0.5057	0.0089
(1)	1.417	0.0972	0.1045	**0.2918**	0.1384	0.4123	0.4102	0.2586
(2)	1.444	**0.0896**	**0.1040**	0.2931	**0.1357**	**0.4030**	0.4116	0.2709
(3)	**1.364**	0.0980	0.1073	0.3055	0.1413	0.4092	**0.4042**	*0.2364*

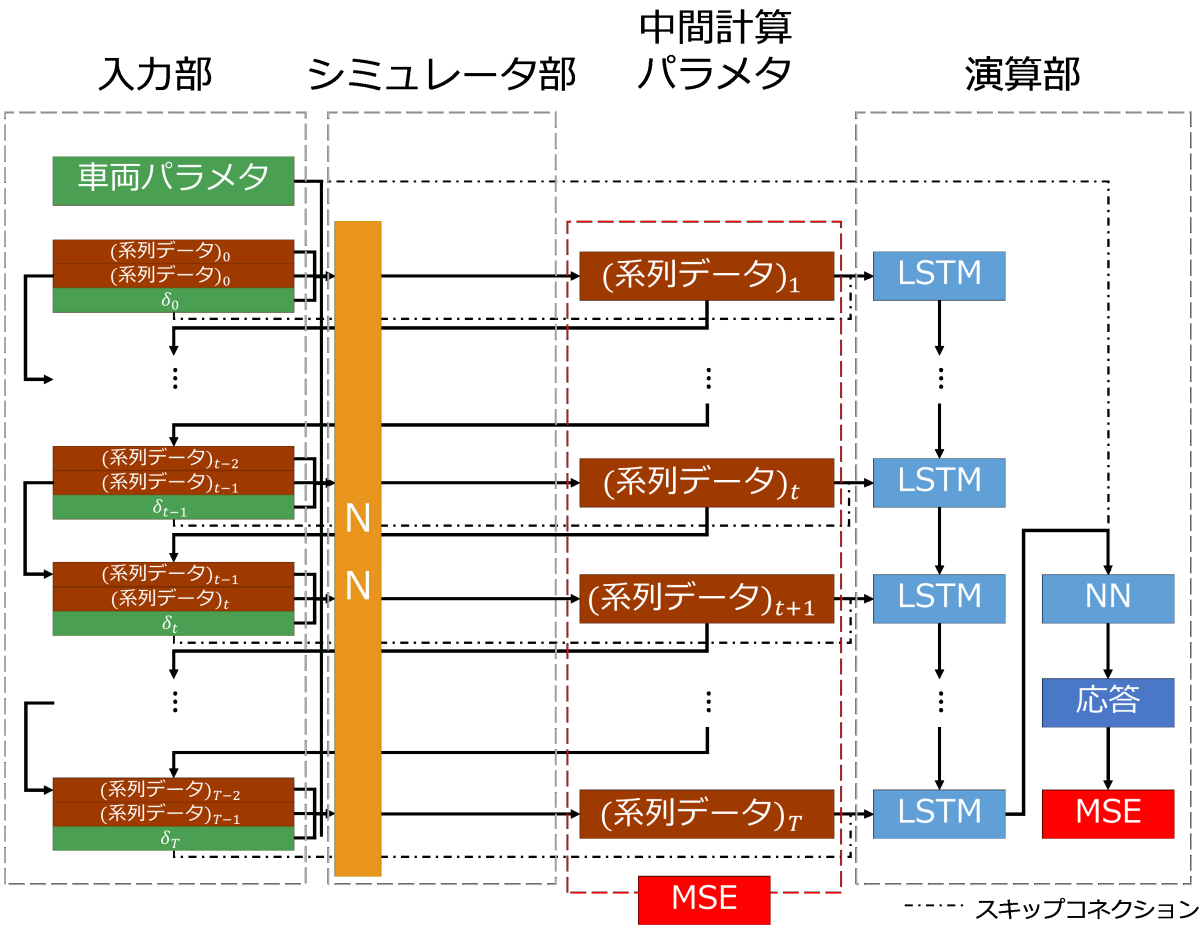

図 16: LSTM をシミュレータ部に用いないモデル

を行ったほうがよいということがわかる．つまり，本タスクに系列学習を取り入れたことによる精度の向上が見られた，ということである．

4.2.6 初期値の仮定の検証

3.2 節ではシミュレーションの初期は緩やかな動作であるという仮定を行い，(3.4) 式のような仮定を行った．この仮定が誤りでないかの確認を行う．この実験では，シミュレーションの始まりを (3.4) とすることと，下の (4.10) 式とした場合について比較実験を行う．

$$\begin{aligned} \hat{\boldsymbol{z}}_1 &= \boldsymbol{z}_1 \\ \hat{\boldsymbol{z}}_2 &= \mathrm{LSTM}(\mathrm{NN}(\boldsymbol{x}_0, \hat{\boldsymbol{z}}_0, \hat{\boldsymbol{z}}_1)) \\ &\vdots \end{aligned} \tag{4.9}$$

この実験の結果が表 10 である．結果を比較すると，$\bar{e}$ は仮定を行ったほうが小さく，精度が高

表 8: シミュレータ部で LSTM を用いないネットワークの構成

	ID	名前	次元
シミュレータ部	1	結合層 + ReLU	512
	2	結合層 + ドロップアウト (0.1)	512
		出力（中間計算パラメタ）	31
演算部	3	結合層 + ReLU	512
	4	LSTM + ReLU	512
	5	結合層 + ドロップアウト（0.1）	512
		出力（応答）	6

表 9: シミュレータ部での LSTM の有無による誤差の比較

モデル	e_1	e_2	e_3	e_4	e_5	e_6	$\bar{e}$	σ_e^2
従来手法	0.6165	0.4915	0.4605	0.5555	0.3492	0.5608	0.5057	0.0089
LSTM を用いないモデル	1.431	**0.0948**	0.1193	0.4296	**0.138**	0.5034	0.4527	0.2597
LSTM を用いたモデル	**1.364**	0.0980	**0.1073**	**0.3055**	0.1413	**0.4092**	**0.4042**	0.2364

かった．本来は仮定を行わず，真値を入れたほうが精度は高くなるはずである．この結果の要因として考えられるのは訓練データの増加によるものであると考えられる．本研究は，4.2.1 節で述べたとおり訓練データ量が不足している．(3.4) の仮定を行うと，行わなかった場合に比べて訓練として用いることができるデータの系列長が 1 ステップ分増加する．このことによって結果がよくなったのではないかと考えられる．

表 10: (3.4) 式の仮定を行った場合と行っていない (4.10) 式の場合の推定誤差の比較

モデル	e_1	e_2	e_3	e_4	e_5	e_6	$\bar{e}$	σ_e^2
従来手法	0.6165	0.4915	0.4605	0.5555	0.3492	0.5608	0.5057	0.0089
(4.10) 式	1.376	**0.0932**	**0.0991**	0.4450	**0.1359**	0.5276	0.4461	0.2425
(3.4) 式（仮定有）	**1.364**	0.0980	0.1073	**0.3055**	0.1413	**0.4092**	**0.4042**	0.2364

第5章　おわりに

本論文を締めるに当たって，本論文のまとめと今後の課題について述べる．

5.1　まとめ

本研究では車輌走行シミュレーションに対して精度が良くなかった，車両パラメタから直接的に応答を求める先行研究に対して精度向上を図るために，系列モデルを導入し，系列学習を行った．シミュレータの中間計算パラメタを系列モデルによって順次計算し，そのパラメタから end-to-end に応答を抽出するモデルを提案した．さらに，ハイブリッド損失・スキップコネクション・スケジュールドサンプリングによって精度向上を図った．実験では本提案手法が従来手法と比べ，推定する応答パラメタの 6 次元のうち，五つに対して精度が向上し，全体として精度が向上したことが確認できた．また，ハイブリッド損失・スキップコネクション・スケジュールドサンプリングによって精度が向上することを確認した．

5.2　今後の課題

- モデルのチューニング

  本論文の提案モデルでは，チューニング箇所が多く，ドロップアウトを加えるなどのチューニングできる箇所もまだ多く残されている．また，ネットワークの全ての部分で層を単層にして行ったため，層の深さについてもチューニングを行わなければならない．このようにモデルの改善できる点が多く存在するため，モデルのチューニングを進めなければならない．

- マルチタスクのためのカリキュラム学習を用いた精度向上

  提案手法の誤差の大きさから，応答パラメタごとに推定難度が異なると考えられる．また，本研究のタスクは，応答パラメタがそれぞれ別のタスクとしてみなすことができるため，マルチタスク学習とみなすことができる．そこで Li らが提案したマルチタスクのためのカリキュラム学習を用いるなど，カリキュラム学習とマルチタスク学習の観点での手法を試すことで精度が向上する可能性がある．

- 汎用的なシミュレータとしてのモデルの構築

  本研究の提案モデルでは，単一種類の車両や単一の環境に対してのモデルとなってしまっている．そこでより汎用的なシミュレータとして用いれるようにモデルを改善するべきである．例えば車両モデルを畳み込みニューラルネットワークによって特徴量にし，それをモデルに

入力するというようなことが考えられる．

- 他のシミュレータや試験に対しての有用性の調査

  他の試験やシミュレータに対しても対応できるかどうか調査するべきであるからデータセットを変えて実験する必要がある．それに付随して，このシミュレータの機械学習化という分野は未発達であり，データセットが少ない．そこで，データ拡張の手法やそもそものデータセットの作成なども必要である．

謝辞

本論文は課題研究の成果をまとめたものです．本研究を行うに当たって多くの方にお世話になりました．後期からの配属先である知能数理研究室では，佐々木裕教授及び三輪誠准教授にお世話になりました．後期からの配属というイレギュラーな事象で短い研究期間でしたが，手厚いご支援によって研究成果をまとめることができました．配属当時は自身も戸惑う部分が多く有りましたが，多くのアドバイスをいただき，研究を進めていくことができました．両先生には深く感謝申し上げます．また，同研究室の先輩方・同級生方のアドバイスで着想を得ることも多く，助けられることが多く有りました．日常生活でも世間話など楽しく過ごすことができました．感謝申し上げます．本研究室は共同研究でした．その共同研究先である，トヨタ自動車株式会社先進技術開発カンパニー車両 CAE 部動的性能 CAE 技術開発室の皆様には深く感謝しています．データの提供や研究に対するアドバイス，本研究の内容についてのご教示など多くのご支援を賜りました．心から感謝申し上げます．前期に配属されていた知能情報メディア研究室では浮田宗伯教授に多大なるご支援を賜りました．研究について何もわからないなか，道を示していただきました．後期に知能数理研究に配属になった後も副指導教員としてご支援いただきました．深く感謝いたします．

参考文献

[1] L'ubor Ladický, SoHyeon Jeong, Barbara Solenthaler, Marc Pollefeys, and Markus Gross. Data-driven fluid simulations using regression forests. *ACM Trans. Graph.*, 34(6):199:1–199:9, October 2015.

[2] Cheng Yang, Xubo Yang, and Xiangyun Xiao. Data-driven projection method in fluid simulation. *Comput. Animat. Virtual Worlds*, 27(3-4):415–424, May 2016.

[3] Jonathan Tompson, Kristofer Schlachter, Pablo Sprechmann, and Ken Perlin. Accelerating eulerian fluid simulation with convolutional networks. In *ICML*, 2017.

[4] Apratim Bhattacharyya, Bernt Schiele, and Mario Fritz. Accurate and diverse sampling of sequences based on a "best of many" sample objective. In *CVPR*, 2018.

[5] National Highway Traffic Safety Administration et al. Laboratory test procedure for dynamic rollover: the fishhook maneuver test procedure. *Washington (DC): US, Department of Transportation*, 2013.

[6] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[8] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[10] Tjalling J Ypma. Historical development of the newton–raphson method. *SIAM review*, 37(4):531–551, 1995.

[11] Jorge J Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis*, pages 105–116. Springer, 1978.

[12] Martin T Hagan and Mohammad B Menhaj. Training feedforward networks with the marquardt algorithm. *IEEE transactions on Neural Networks*, 5(6):989–993, 1994.

[13] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[15] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017.

[16] Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *NeurIPS*, 2018.

[17] Jie Xu, Lin Zhao, Shanshan Zhang, Chen Gong, and Jian Yang. Multi-task learning for object keypoints detection and classification. *Pattern Recognition Letters*, 08 2018.

[18] Changsheng Li, Junchi Yan, Fan Wei, Weishan Dong, Qingshan Liu, and Hongyuan Zha. Self-paced multi-task learning. In *AAAI*, pages 2175–2181, 2017.

[19] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179, 2015.