自主課題研究 最終レポート

RNN を用いた POMDP 環境での倒立振り子問題

金沢大学 理工学域
電子情報通信学類 3 年

清水 翔仁

令和 2 年 2 月 17 日

1章　はじめに

強化学習は，試行錯誤から学習する点で様々な問題に適用しやすいため，近年盛んに研究されている．DQN(Deep Q-Network) は，Q 学習にディープラーニングを用いた強化学習のモデルで，ゲーム画面を入力した Atari 2600 の様々なゲームに対して人間より良い性能を実現した．しかし，DQN の入力はエージェントが観察した最近の 4 つのフレームのみを用いるため，意味がある行動をするために 4 ステップより昔の情報を必要とするような環境に対しては性能が低下する可能性がある．この欠点を解決するために DRQN(Deep Recurrent Q-Network) が提案されている．DRQN は，DQN に循環神経網の一種である LSTM を組み合わせることで過去の情報を扱う．さらに LSTM はより長期的な情報を考慮することで，現実で起こりうる不完全な情報にも対応できるため，POMDP(Partially Observable Markov Decision Process) の環境でより効果を発揮することができる．

しかし，DRQN は学習の時に LSTM の初期状態をゼロベクトルで初期化するため，学習時に用いた情報のタイムステップより長い期間に対して学習することが難しいという欠点がある．そこで本研究では，LSTM の初期状態を与える方法を複数用意し，それぞれの実行結果についての考察を行う．

2 章　原理

2.1　POMDP(Partially Observable Markov Decision Process)

状態遷移が確率的に生じる動的システムの確率モデルを MDP という．遷移する過程において，将来状態の条件付き確率分布が，現在状態のみに依存し，過去のいかなる状態にも依存しない．(マルコフ性)

各時刻において過程 (process) はある状態 (state) を取り，意思を決定するエージェントはその状態において利用可能な行動 (action) を任意に選択する． その後過程はランダムに新しい状態へと遷移し，その際にエージェントは状態遷移に対応した報酬 (reward) を受けとる．よって MDP は以下の要素で構成される．

$$
\begin{aligned}
&\text{行動集合} \ : \mathcal{A} = \left\{a^{(1)}, a^{(2)}, \cdots, \right\} \\
&\text{状態集合} \ : S = \left\{s^{(1)}, s^{(2)}, \cdots, \right\} \\
&\text{遷移関数} \ : T_{ijk} = \Pr\left(s_{t+1} = s^{(j)} \mid s_t = s^{(i)}, a_t = a^{(k)}\right) \\
&\text{報酬関数} \ : r = g(s, a) \\
&\text{初期状態確率} \ : p_0 = \Pr\left(s_0\right)
\end{aligned}
\tag{2.1}
$$

POMDP はエージェントの状態観測に不確実性を付加させることにより MDP を拡張したものである．

2.2　Q 学習 (Q-learning)

Q 学習は強化学習における手法の一種である．Q 学習は MDP において全ての状態が十分にサンプリングできるようなエピソードを無限回試行した場合，最適な評価値に収束することがわかっている．Q 学習では実行するルールに対しそのルールの有効性を示す Q 値という値を持たせ，エージェントが行動するたびにその値を更新する．ここでいうルールとはある状態とその状態下においてエージェントが可能な行動を対にしたものである．例えばエージェントの現在の状態を s_t とし，この状態で可能な行動が a，b，c，d の 4 通りあるとする．このとき，エージェントは 4 つの Q 値，$Q(s_t, a)$，$Q(s_t, b)$，$Q(s_t, c)$，$Q(s_t, d)$ を元に行う行動を決定する．行動の決定方法は理論上では無限回数試行するならランダムでも Q 値は収束するが，現実には収束を早めるため，なるべく Q 値の大きな行動が高確率で選ばれるように行う．Q 値の計算式を式 (2.2) に示す．

$$
Q\left(s_t, a_t\right) = (1-\alpha) Q\left(s_t, a_t\right) + \alpha \left(r_{t+1} + \gamma \max_{a_{t+1}} Q\left(s_{t+1}, a_{t+1}\right)\right) \tag{2.2}
$$

ここで γ は割引率といい，将来の価値をどれだけ割り引いて考えるかのパラメータである．

2.3 DQN(Deep Q-Network)

Q 値を最大化させる関数である最適行動価値関数を，ニューラルネットワーク (NN) を使った近似関数で求める手法．状態を NN の入力にし，出力層の各ノードが，各行動の行動価値を出力するようにする．

強化学習において与えられるデータは時系列的に連続したものになっており，データ間に相関が出てしまうためバラバラにする必要がある．これを Experience Replay という．

2.4 RNN(Recurrent Neural Network)

RNN とは，Deep Learning の手法の一種であり，時系列データの分析に特化している．図 2.1 に，RNN の模式図を示す．ここで，x_t を時刻 t における RNN の入力，h_t を時刻 t における RNN の出力とする．

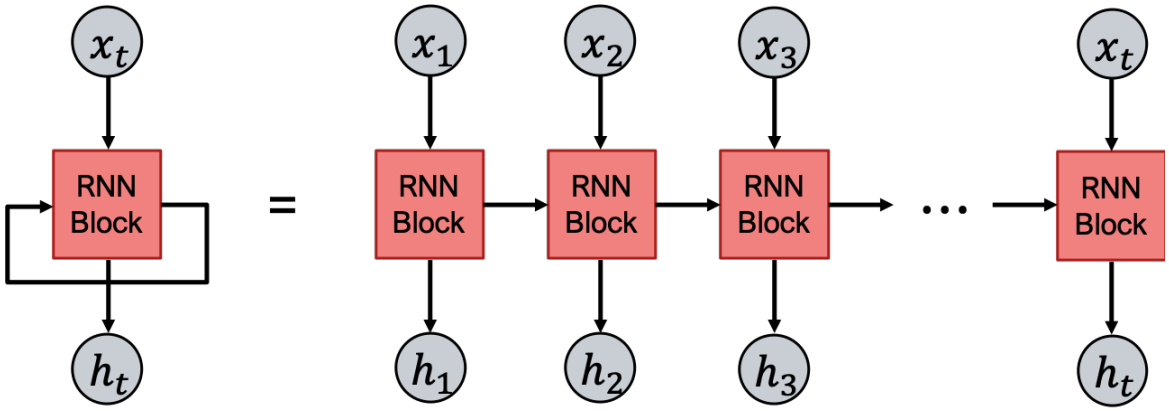

図 2.1: 展開された RNN

図 2.1 のように，RNN は内部にループ構造を持っている．これにより，前のステップでの分析結果を記憶し，データの時系列を理解することができる．しかし，RNN には長期依存性問題という欠点がある．長期依存性問題とは，記憶するステップ数が膨大になると計算が爆発するという問題である．そのため，現在単純な RNN はあまり使用されていない．

2.5 LSTM(Long Short Term Memory)

LSTM とは，RNN の長期依存性問題を解決した手法である．図 2.2 に RNN の内部を，図 2.3 に LSTM の内部を示す．このとき，σ は 0〜1 を出力する関数，tanh は-1〜1 を出力する関数とする．

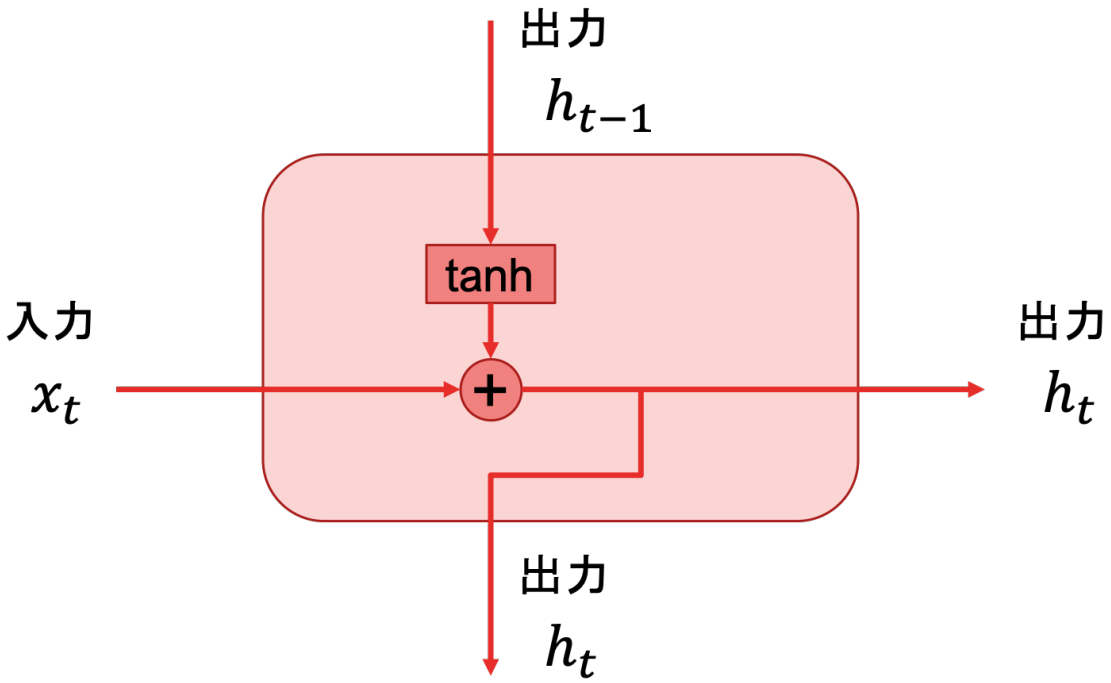

図 2.2: RNN の内部

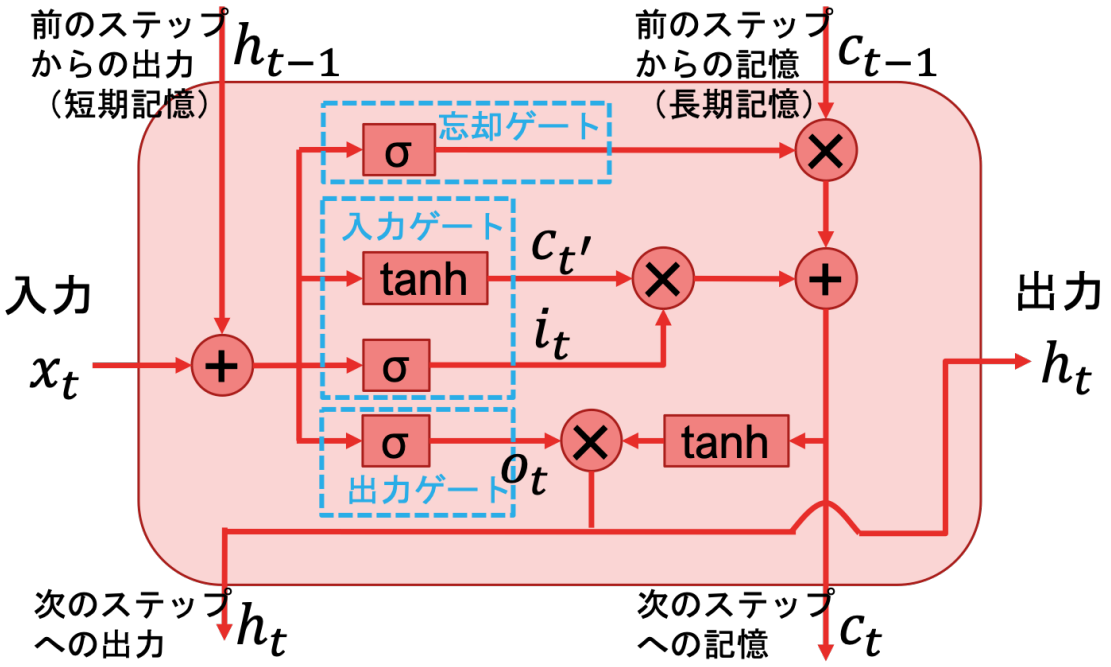

図 2.3: LSTM の内部

図 2.2，図 2.3 より，RNN は単一の tanh 関数という非常に単純な構造に比べ，LSTM は 4 つの関数を含む複雑な構造をしている．

ここで，図 2.3 の LSTM 内にデータが流れる手順について説明する．

1. 前のステップからの出力 h_{t-1} と入力 x_t が合流する．合流した信号はコピーされて 4 つのラインに分岐する．
2. 一番上のラインの忘却ゲートでは，前のステップからの記憶一つ一つに対して，σ 関数からの 0〜1 の値によって情報の取捨選択を行う．このとき 1 は情報を全て残し，0 は全て捨てる．これにより，不要と思われる情報を捨てることで計算の爆発を防ぐ．
3. 入力ゲートにおいて，前のステップからの出力 h_{t-1} と入力 x_t の合算を長期保存用に変換した上で，どの信号をどのくらいの重みで記憶に保存するか制御する．これは 2 つの手順で処理する．
   (a) tanh 関数を用いて，入ってきた情報の情報量を削減し，必要な情報だけに変換された $c_{t'}$ が出力する．
   (b) σ 関数の出力 i_t によって，h_{t-1} を考慮して入力 x_t の重みを調整する．
4. 出力ゲートにおいて，上記の処理で取捨選択された長期記憶 c_t の中で，短期記憶 h_t に関する部分のみを出力する．これも 2 つの手順で処理する．
   (a) 前のステップからの記憶 c_{t-1} と，入力 x_t を変換した短期記憶 $c_{t'}$ を合算し，長期記憶 c_t として出力する．これは，それぞれ既に忘却ゲートおよび入力ゲートで取捨選択が行われている．
   (b) tanh 関数に c_t を入力したものに対し，σ 関数からの 0〜1 の値 o_t によって情報の取捨選択を行う．

3 章　実装方法

本章では，開発環境や精度向上のために行ったことについて説明する．

3.1　開発環境

本研究の開発環境を表 3.1 に示す．

表 3.1: 開発環境

OS	macOS Catalina 10.15.2
CPU	Intel(R) Core i7-5650U CPU@2.20GHz
プログラミング言語	Python 3.7.4
ニューラルネットワークライブラリ	keras 2.3.1
デバッグツール	TensorBoard 2.1.0

3.2　学習データセット

今回の学習データセットは，気象庁ホームページ [3] にて公開されている気象観測データをダウンロードして使用する．ダウンロードするにあたって指定した条件を以下に示す．

- データ形式:CSV
- 期間:1998/1/1 - 2019/12/31
- 地点:長野市
- 特徴量:日平均現地気圧，日平均気温，日最低気温，日最高気温，降水量の日合計，日照時間，日平均風速，日最大風速，日平均相対湿度，日最小相対湿度，日平均雲量

次に，実際にダウンロードしたデータの例を表 3.2 に示す．表 3.2 より，この状態では気圧の値のみが明らかに桁が大きいため，全体に正規化の処理を施す．

表 3.2 の値を正規化したものを表 3.3 に示す．表 3.3 より，全ての値が 0〜1 の間に収まっており，大小関係も変化していないことがわかる．

表 3.2: 学習データの例

日付	気圧 [hPa]	平均気温 [°C]	最高気温 [°C]	最低気温 [°C]
1998/1/1	965.8	0.4	5.5	-3.9
1998/1/2	968.3	3.2	8.0	0.0
1998/1/3	969.9	2.3	9.8	-3.0
1998/1/4	960.8	3.2	9.7	-1.1

表 3.3: 正規化後の学習データ

日付	気圧	平均気温	最高気温	最低気温
1998/1/1	0.618	0.189	0.218	0.205
1998/1/2	0.670	0.263	0.277	0.303
1998/1/3	0.704	0.239	0.320	0.227
1998/1/4	0.514	0.263	0.318	0.275

4 章　結果

ここでは，隠れ層の数によって予測の精度がどの程度向上するのかを調べるため，隠れ層の数が 1 つの場合，2 つの場合，3 つの場合の 3 パターンについてプログラムを実行する．

まず隠れ層が 1 つの場合についての結果を図 4.1，図 4.2 に示す．図 4.1 より，精度が右肩上がりに上昇し，90% 近くに達していることがわかる．また図 4.2 より，損失が右肩下がりに減少し，値が 0.25 ほどになっていることがわかる．

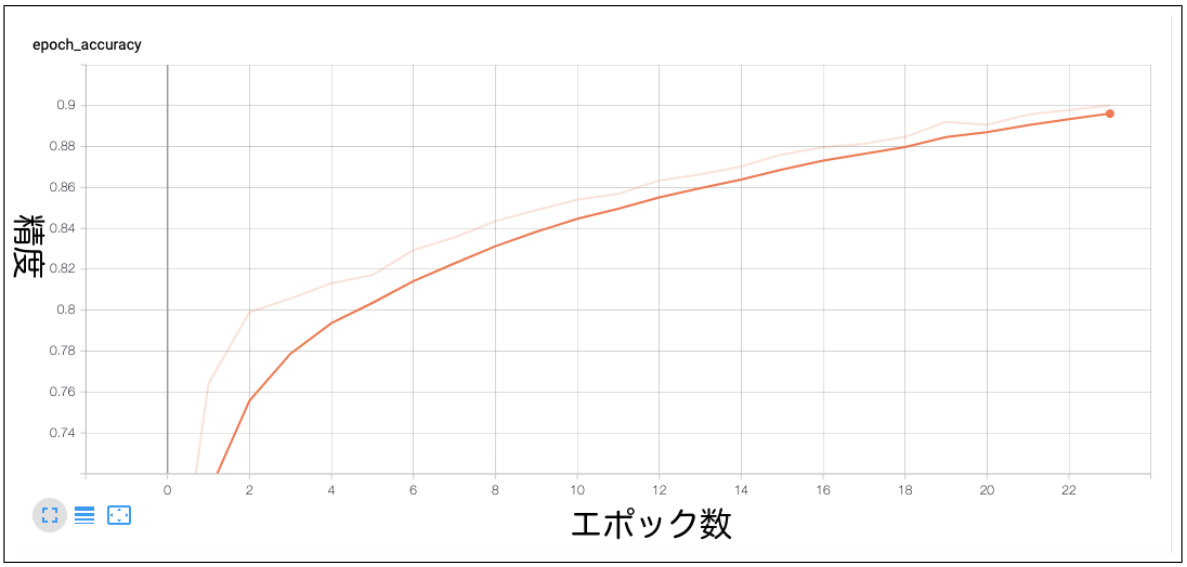

図 4.1: 隠れ層が 1 つの場合の精度

次に隠れ層が 2 つの場合についての結果を図 4.3，図 4.4 に示す．図 4.3 より，精度が 99% に達し図 4.1 よりも高い精度が得られている．また図 4.4 より，損失が 0 に近い値になり図 4.2 よりも減少していることがわかる．このことから，隠れ層の数を増やすことが精度向上に寄与していると考えられる．

隠れ層が 3 つの場合についての結果を図 4.5，図 4.6 に示す．図 4.5 より，エポック数が 7 の時点で精度の上昇が頭打ちになり，エポック数 8 で減少していることがわかる．また図 4.6 より，下がっていた損失がエポック数 7 で上昇していることがわかる．このことから，隠れ層を増やしたことによりニューラルネットワークが複雑になり，過学習が発生していると考えられる．また，以上の結果から隠れ層の数は 2 つが適当であると考えられる．

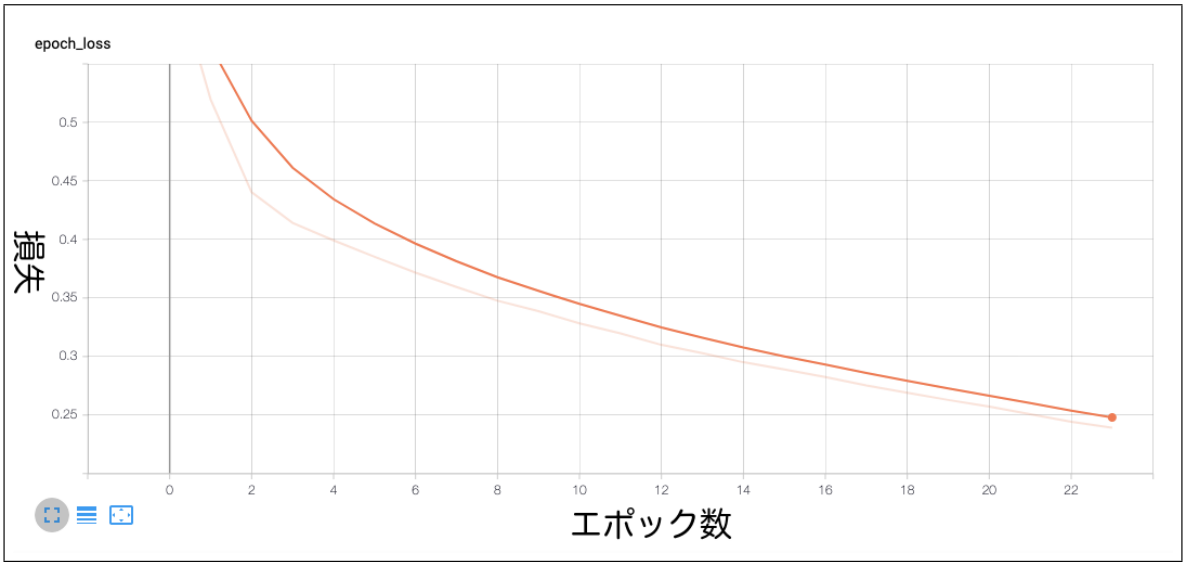

図 4.2: 隠れ層が 1 つの場合の損失

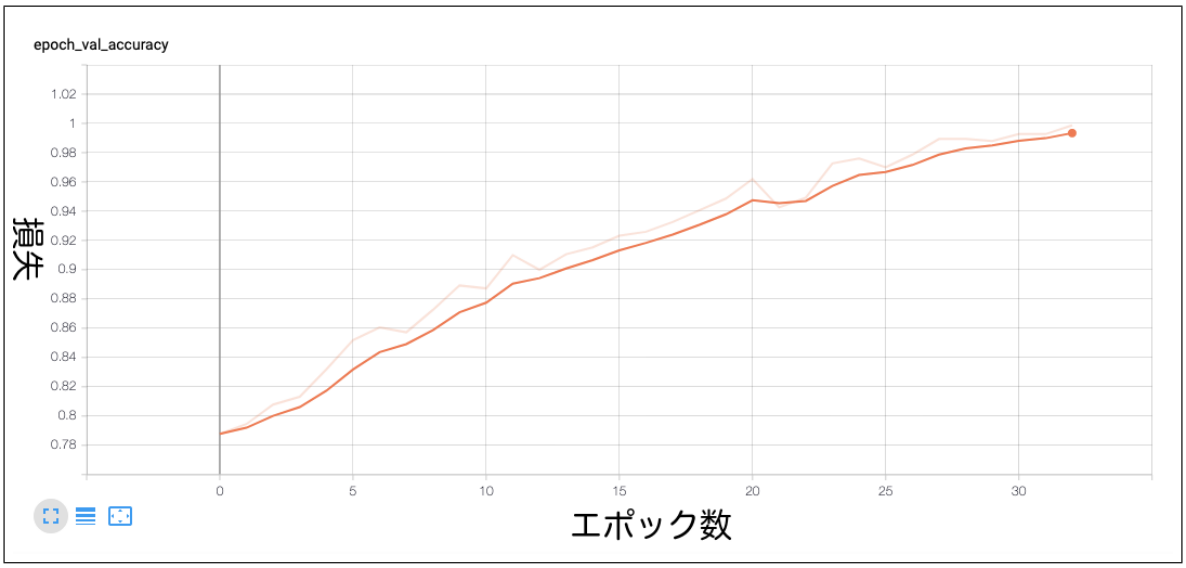

図 4.3: 隠れ層が 2 つの場合の精度

隠れ層の数を 2 つに固定し，プログラムを 10 回実行した際の精度を表 4.1 に示す．表 4.1 より，予測の精度は約 94% となった．よって，かなり高い精度で降水量の有無を予測できていると考えられる．

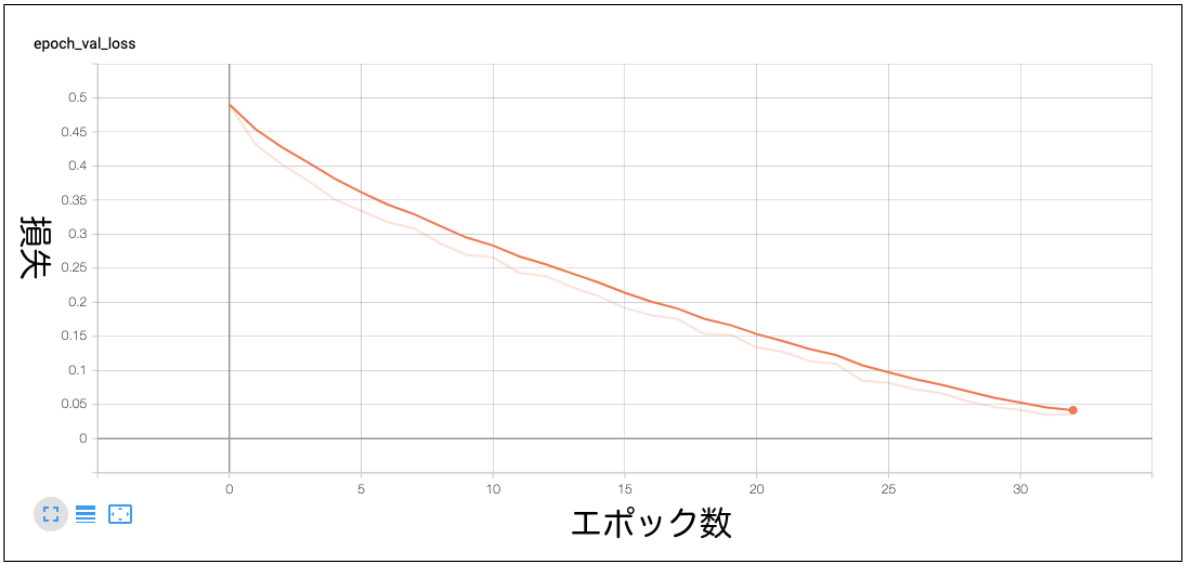

図 4.4: 隠れ層が 2 つの場合の損失

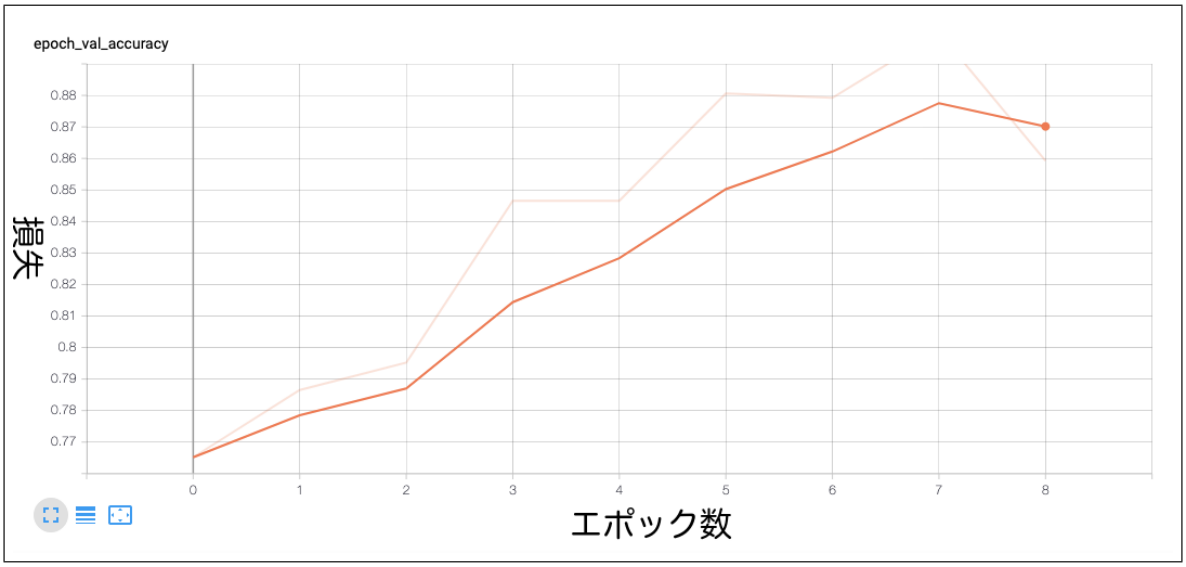

図 4.5: 隠れ層が 3 つの場合の精度

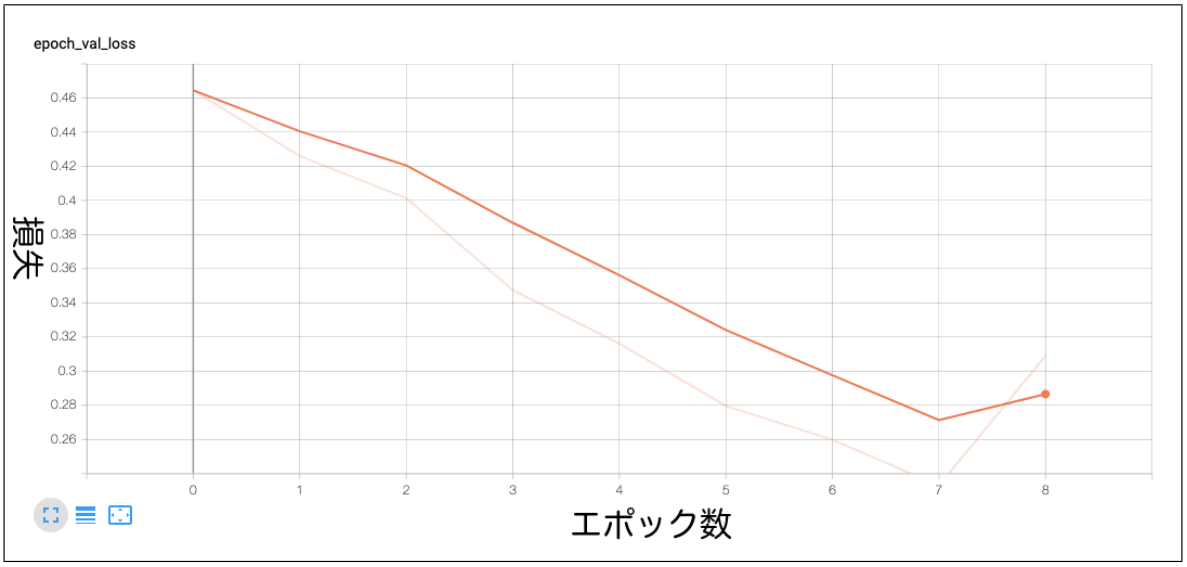

図 4.6: 隠れ層が 3 つの場合の損失

表 4.1: 予測精度

番号	精度
1	0.9248
2	0.9845
3	0.8873
4	0.9028
5	0.9521
6	0.9578
7	0.9560
8	0.9601
9	0.9380
10	0.9091
平均	0.93725

5章　まとめ

本研究の目標である，過去の気象観測データから翌日の降水量の有無を予測することができた．また研究を進める中で，機械学習を行うプログラムの，実装方法を学ぶことができた．

改善点としては，現在日別に出力している予測結果を，より詳細な 1 時間毎の結果に拡大すること．そして降水の有無ではなく，どれくらい雨が降るか，というような降水量の予測を行うことが挙げられる．

参考文献

[1] 鬼頭 葉子: 技術者の倫理, ナカニシヤ出版, 2018.

[2] 気象庁 — 数値予報課報告・別冊第 64 号 (令和 2 年 2 月 16 日現在): https://www.jma.go.jp/jma/kishou/books/nwpreport/64/chapter1.pdf

[3] 気象庁 — 過去の気象データ・ダウンロード (令和 2 年 2 月 16 日現在): https://www.data.jma.go.jp/gmd/risk/obsdl/index.php

[4] Antonio Gulli・Sujit Pal(著)，大串正矢・久保隆宏・中山光樹 (訳): 直感 Deep Learning, 株式会社オライリー・ジャパン, 2019.

[5] Sebastian Raschka・Vahid Mirjalili(著)，株式会社クイープ (訳): [第 2 版]Python 機械学習プログラミング, 株式会社インプレス, 2018.

[6] TensorBoard(令和 2 年 1 月 20 日現在): https://www.tensorflow.org/tensorboard