令和 5 年度　　修士論文

格子ボルツマン法を組み込んだ
深層学習モデルによる日本近辺の風速予測

指導教諭 澤田 秀之 教授

2024 年 2 月 5 日提出

早稲田大学　先進理工学部
応用物理学科
澤田秀之研究室 修士 2 年

5322A083-2　　山倉　拓也

目次

第 1 章

はじめに

1.1 研究背景

本研究の背景には，ニューラルネットワークを用いた深層学習と格子ボルツマン法 (Lattice Boltzmann Method, LBM) の二つの重要な要素がある．一方の深層学習は機械学習の一分野であり，複数の隠れ層を持つニューラルネットワークを使って複雑なパターンを学習する手法である [1]．近年盛んに研究されていて応用先の一つに気象予測があり [2]，例えば CNN(Convolutional Neural Network)[3] と LSTM(Long Short-Term Memory)[4] を用いた正方形領域での風速予測がされている [5]．他方で，格子ボルツマン法は 1990 年代以降に発展した比較的新しい数値流体力学の手法であり，個々の粒子のふるまいを扱うのではなく，格子状に分割した空間内で各格子点上の離散化された速度分布関数を解くものである [6]．この手法の長所として並列計算と相性がよいことが挙げられ，GPU や TPU のようなプロセッサ上で効率的に計算することができる [7]．これはニューラルネットワークにも共通する特性である [8]．

1.2 先行研究

1.2.1 CNN と LSTM を用いた正方形領域での風速予測

Chen ら (2021) は，CNN と LSTM を用いたモデルによる正方形領域での風速予測を行った [5]．彼らのモデル概要図を図 1.1 に示す．この図において，エンコーダとデコーダの部分に CNN が用いられており，これにより正方形領域における格子点上の風速の空間的なパターンを学習している．また，エンコーダの出力を LSTM に入力し，更にその出力をデコーダに入力している．この LSTM のユニットにより，時間的なパターンも学習している．

このモデルを使って，彼らは図 1.2 に示すようにアメリカのインディアナ州内部で，2km 間隔の 10×10 格子点上で 2 時間先の風速予測を行い，単に ANN(Artificial Neural Network)[9] や LSTM のみを用いたモデルに比較して精度が向上することを示した．具体的には，全体の風速予測の平均絶対誤差 (Mean Absolute Error, MAE) が 0.35m/s 減少し，これは Persistence モデル，ANN のみのモデル, LSTM のみのモデルの結果に比べてそれぞれ 32.7%, 28.8%, 18.9% 低い値で

あると主張している．ここで，Persistence モデルとは，風速の時間変化を考慮せず，直前の風速をそのまま予測値とするモデルである．

1.2.2 Physics-Informed Neural Networks

Raissi(2020) らは，物理学の知識をニューラルネットワークに組み込むことで，データが少ない場合でも高精度な予測を行うことができる Physics-Informed Neural Networks(PINNs) というモデルを提案した [10]．このモデルは流体のシミュレーションに使われることが多く [11]，ここでもその用途を想定する．彼らの提案したモデル概要図を図 1.3 に示す．この図が示すように，彼らのモデルは座標 x,y,z と時刻 t を入力とし，それに対応する流体のパッシブスカラーの濃度 c, 流体の速度 u,v,w，そして流体の圧力 p を出力する．なおパッシブスカラーとは，流体の流れによって変化するが流体の流れに影響を与えない物理量のことであり，例えば流体に溶けた微量な染料の濃度などが挙げられる [12]．

彼らのモデルは，損失関数 (2.1.1 項参照) に物理的な微分方程式からどれだけモデルの出力が乖離しているかを表す項を組み込むことで，物理法則を満たすように学習をする．図 1.3 中では

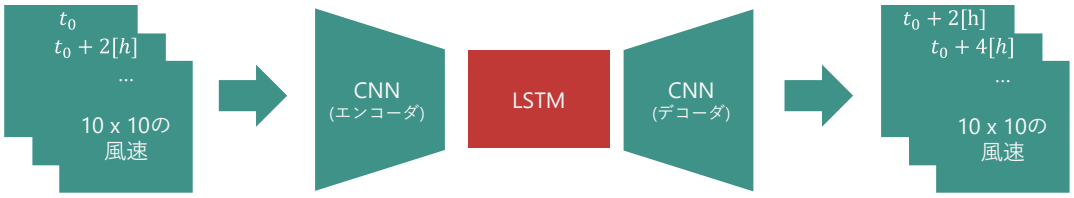

図 1.1　Chen ら (2021) のモデル概要図 [5]

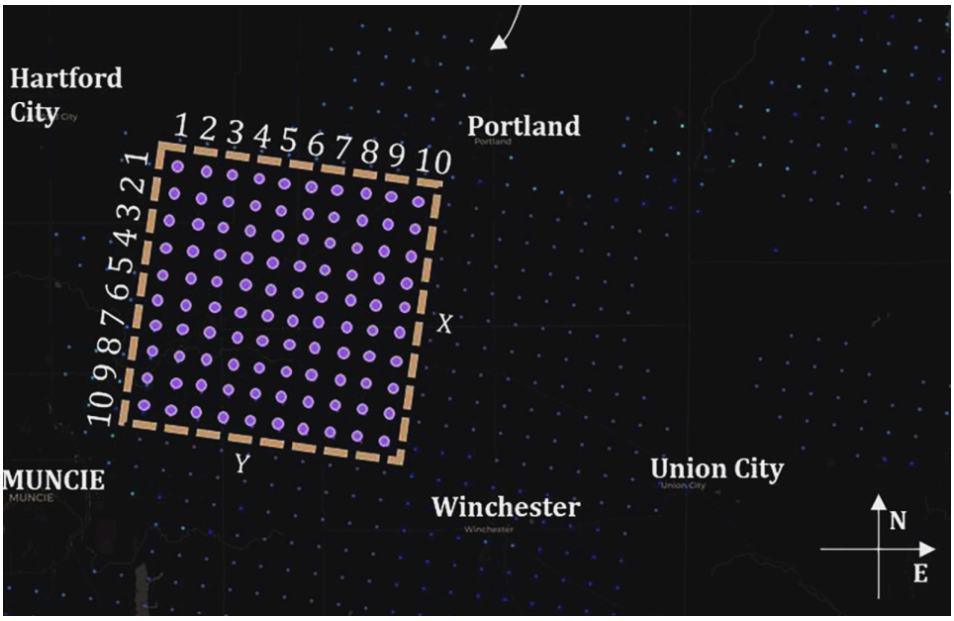

図 1.2　Chen ら (2021) が風速予測を行った範囲 [5]

e_1,e_2,e_3,e_4,e_5 がこの項に当たる．このとき，物理量の座標微分や時間微分は，2.1.2 項に述べるように自動微分を使って導くことができる．実測データが少ない場合でも，それ以外の座標と時刻を入力した場合の損失が小さくなるようにすれば，モデルは物理法則に従うと彼らは主張している．

1.3 研究目的

本研究では 1.2.1 項で述べた Chen ら (2021) の研究と同様に，長方形領域における格子点上での風速予測を行うが，ニューラルネットワークの構造に LBM を組み込むことで，LBM の持つ並列計算と相性の良さを活かしつつ，精度の向上を図る．また 1.2.2 項で述べた Raissi ら (2020) の研究とは対照的に，大気の風速予測のようにデータセットが十分あるが外力や拘束条件が不明または特定が困難な場合にも，物理法則を利用できるモデルを提案することも目的としている．

1.4 論文構成

本論文では，第 2.1 節で深層学習の，第 2.2 節で格子ボルツマン法の数理的な原理を解説し，それらを踏まえて第 3 章で提案モデルでは格子ボルツマン法をどのように深層学習モデルに組み込んだのかを解説する．第 4 章では，提案モデルの有効性を検証するため，日本近辺の風速予測を行い，従来研究のモデルと比較検討をする．

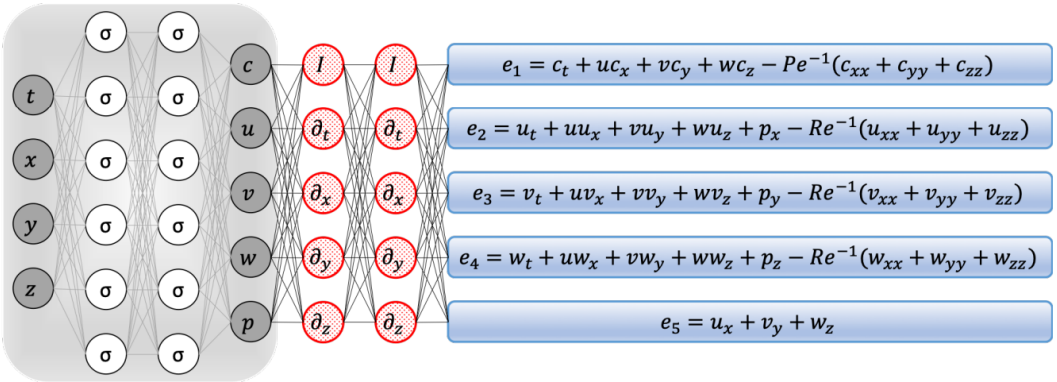

図 1.3　Raissi ら (2020) のモデル概要図 [10]

第 2 章

原理

2.1 深層学習

深層学習は機械学習の一分野であり，複数の隠れ層から成るニューラルネットワークを使用して高度なパターン認識や特徴抽出を行う手法である [1]．この章では最も単純で典型的な深層学習のアーキテクチャを説明する．

図 2.1 に示す通り，深層学習は入力層，隠れ層，出力層から成るニューラルネットワークで表される．入力層は入力データを受け取り，隠れ層は入力層から出力を受け取り，出力層は隠れ層の出力を受け取る．隠れ層は複数存在し，それぞれの隠れ層は前の隠れ層の出力を受け取る．ここで，各層は複数のニューロンからなり，各ニューロンは入力された実数値に活性化関数を適用した値を出力する．図 2.1 中の破線矢印が活性化関数に対応する．そしてその入力値は，前の層のニューロンの出力に重みをかけたものの和にバイアスを加えた値である．これは図 2.1 中の実線矢印に対応する．十分な隠れ層を持ち，重みが適切に調節されたニューラルネットワークは任意の関数を近似することができることが知られている [13].

2.1.1 ニューラルネットワークの計算原理

より詳しく原理を説明する．合計で m 層になるニューラルネットワークを考える．まず，入力層のニューロンの数を $n^{(1)}$ とし，第 j $(1 \leq j \leq n^{(1)})$ ニューロンへの入力を $y_j^{(1)} \in \mathbb{R}$，ニューロンからの出力を $z_j^{(1)} \in \mathbb{R}$ とする．$y_j^{(1)}$ は入力データの第 j 成分を意味する．また，$z_j^{(1)}$ は $y_j^{(1)}$ に活性化関数 $f^{(1)} : \mathbb{R} \to \mathbb{R}$ を適用した値である．つまり，

$$z_j^{(1)} = f^{(1)}(y_j^{(1)}) \tag{2.1}$$

である．次に，中間層または出力層である第 k $(2 \leq k \leq m)$ 層のニューロンの数を $n^{(k)}$ とし，第 j $(1 \leq j \leq n^{(k)})$ ニューロンへの入力を $y_j^{(k)} \in \mathbb{R}$，ニューロンからの出力を $z_j^{(k)} \in \mathbb{R}$ とすると，これらは以下のように表される．

$$y_j^{(k)} = \sum_{i=0}^{n^{(k-1)}} w_{ij}^{(k)} z_i^{(k-1)} \tag{2.2}$$

$$z_j^{(k)} = f^{(k)}(y_j^{(k)}) \tag{2.3}$$

ここで，$w_{ij}^{(k)} \in \mathbb{R}$ は第 $k-1$ 層の第 i ニューロンから第 k 層の第 j ニューロンへの重み，$f^{(k)} : \mathbb{R} \to \mathbb{R}$ は活性化関数である．ただし，$f^{(m)}$ は恒等関数とする．活性化関数は通常非線形関数であり，シグモイド関数や ReLU 関数などが用いられる．またここでは $z_0^{(k-1)} = 1$ とすることで，バイアス $w_{0j}^{(k)}$ を導入する．

以上から，ニューラルネットワークに $y_1^{(1)}, \ldots, y_{n^{(1)}}^{(1)}$ を入力すると式 (2.1), (2.2), (2.3) によって順次計算が行われ，$z_1^{(m)}, \ldots, z_{n^{(m)}}^{(m)}$ が出力されることがわかる．続いて，これをどのように目的の出力に近づけるかを考える．

目的の出力を $\hat{z}_0^{(m)}, \ldots, \hat{z}_{n^{(m)}}^{(m)}$ とする．このとき，出力層のニューロンの出力と目的の出力の誤差を表す二乗損失関数は次のように表される（これは単に損失とも呼ばれる）．

$$E = \frac{1}{2} \sum_{j=1}^{n^{(m)}} \left(z_j^{(m)} - \hat{z}_j^{(m)} \right)^2 \tag{2.4}$$

これを最小化するように各層の重みを調節することで，ニューラルネットワークは目的の出力に近づく．ここでは最も単純な勾配降下法を用いる．まず，隠れ層及び出力層に接続する重み $w_{ij}^{(k)}$ $(2 \leq k \leq m)$ の誤差 E に対する勾配は $\partial E / \partial w_{ij}^{(k)}$ で表され，これにより重みを次のように更新すると誤差が減少することが分かる．

$$w_{ij}^{(k)} \leftarrow w_{ij}^{(k)} - \eta \frac{\partial E}{\partial w_{ij}^{(k)}} \tag{2.5}$$

ここで，η は学習率と呼ばれる 0 以上の実数である．以下では $\partial E / \partial w_{ij}^{(k)}$ を求めることを考える．そのために準備としてひとつ記号を定義する．$\delta_j^{(k)}$ を第 k 層の第 j ニューロンの誤差といい，以下

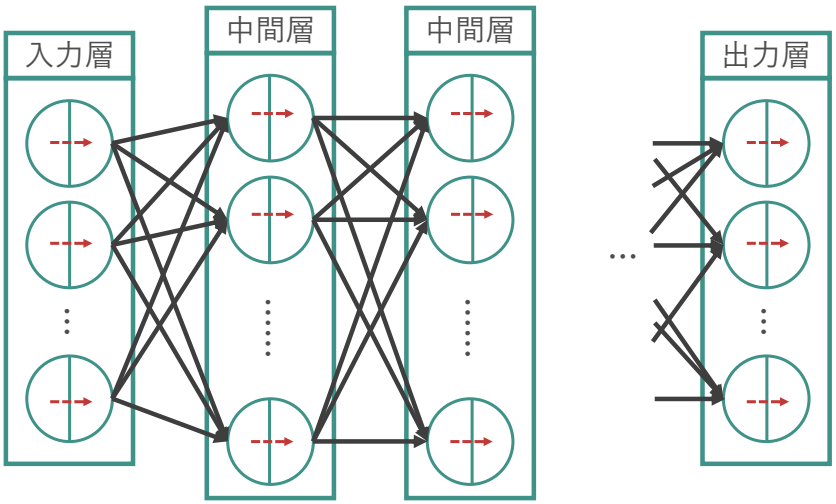

図 2.1　深層学習のアーキテクチャ

で定義する．

$$\delta_j^{(k)} = \frac{\partial E}{\partial u_j^{(k)}} \tag{2.6}$$

これを用いて $\partial E/\partial w_{ij}^{(k)}$ を表し，最後に誤差を求める．

$2 \leq k \leq m$ とする．式 (2.3), (2.6) より，$\partial E/\partial w_{ij}^{(k)}$ は次のように表される．

$$\frac{\partial E}{\partial w_{ij}^{(k)}} = \frac{\partial E}{\partial u_j^{(k)}} \frac{\partial u_j^{(k)}}{\partial w_{ij}^{(k)}} = \delta_j^{(k)} \frac{\partial u_j^{(k)}}{\partial w_{ij}^{(k)}} \tag{2.7}$$

そして，式 (2.2) より，$\partial u_j^{(k)}/\partial w_{ij}^{(k)}$ は次のように表される．

$$\frac{\partial u_j^{(k)}}{\partial w_{ij}^{(k)}} = \frac{\partial \left(\sum_{i=0}^{n^{(k-1)}} w_{ij}^{(k)} z_i^{(k-1)} \right)}{\partial w_{ij}^{(k)}} = z_i^{(k-1)} \tag{2.8}$$

したがって，式 (2.7), (2.8) より，

$$\frac{\partial E}{\partial w_{ij}^{(k)}} = \delta_j^{(k)} z_i^{(k-1)} \tag{2.9}$$

となる．

最後に誤差を求める．まず，出力層の誤差 $\delta_j^{(m)}$ は式 (2.6), (2.4) より次のように表される．

$$\delta_j^{(m)} = \frac{\partial E}{\partial u_j^{(m)}} = \frac{\partial E}{\partial z_j^{(m)}} = \frac{\partial \left(\frac{1}{2} \sum_{j=1}^{n^{(m)}} \left(z_j^{(m)} - \hat{z}_j^{(m)} \right)^2 \right)}{\partial z_j^{(m)}} = z_j^{(m)} - \hat{z}_j^{(m)} \tag{2.10}$$

続いて，$2 \leq k \leq m-1$ に対して，第 k 層の誤差 $\delta_j^{(k)}$ は次のように表される．

$$\begin{aligned} \delta_j^{(k)} &= \frac{\partial E}{\partial u_j^{(k)}} = \sum_{i=1}^{n^{(k+1)}} \frac{\partial E}{\partial u_i^{(k+1)}} \frac{\partial u_i^{(k+1)}}{\partial u_j^{(k)}} \\ &= \sum_{i=1}^{n^{(k+1)}} \delta_i^{(k+1)} \frac{\partial u_i^{(k+1)}}{\partial z_j^{(k)}} \frac{\partial z_j^{(k)}}{\partial u_j^{(k)}} \\ &= \sum_{i=1}^{n^{(k+1)}} \delta_i^{(k+1)} w_{ji}^{(k+1)} f^{(k)\prime}(y_j^{(k)}) \end{aligned} \tag{2.11}$$

以上より，式 (2.10), (2.11) を用いて $\delta_j^{(k)}$ を順次計算し，さらに式 (2.9) を用いて $\partial E/\partial w_{ij}^{(k)}$ を順次計算することで，式 (2.5) によって重みを更新することができる．

2.1.2 自動微分

2.1.1 項では，ニューラルネットワークの重みを更新するために誤差を逐次的に求めていく必要があることを述べた．しかし，ニューラルネットワークのモデルが複雑化するに従ってこの数式を

手作業で計算することは困難になってくる．そこで，自動微分と呼ばれる手法を用いる．一般的にコンピュータ上で表される関数は，基本的には四則演算や三角関数，指数関数などの初等関数の合成で表される [14]．自動微分では，このような基本的な関数の微分をあらかじめ定義しておき，合成関数の微分を関数の微分の積で表すことで計算する [14]．PyTorch[15] や TensorFlow[16] などの深層学習フレームワークでは，この自動微分が実装されているため複雑なモデルでも容易に実装することができる．

2.2 格子ボルツマン法

格子ボルツマン法 (LBM, Lattice Boltzmann Method) は数値流体力学の一手法である．流体を有限種類の速度を持つ仮想粒子の集合とみなし，その分布関数を時空間で離散化したものに対して並進と衝突と呼ばれる操作を逐次施すことで流体の動きを表現する [6]．この章では，LBM の基本的な原理を説明する．

等温場の格子流体モデルとして，ここでは D2Q9 モデルを考える．D2Q9 モデルは，仮想粒子の速度ベクトルが以下の式に示す 9 種類である二次元空間格子モデルである．

$$\boldsymbol{v} = (\pm 1, \pm 1), (\pm 1, 0), (0, \pm 1), (0, 0) \tag{2.12}$$

ただし，複号任意である．これは図 2.2 に示すとおり，速度ベクトルは格子の中心からのオフセットで表されることを意味している．この下で，分布関数 $f(\boldsymbol{x}, \boldsymbol{v}, t)$ は無次元座標 $\boldsymbol{x} \in \mathbb{R}^2$，無次元時刻 $t \in \mathbb{R}$ における速度 $\boldsymbol{v}$ の粒子数を表す．すると，これは任意の $\boldsymbol{v}$ について以下の離散ボルツマン方程式に従う [17].

$$\mathrm{Sh}\frac{\partial f(\boldsymbol{x}, \boldsymbol{v}, t)}{\partial t} + \boldsymbol{v} \cdot \nabla f(\boldsymbol{x}, \boldsymbol{v}, t) = \frac{1}{\epsilon}\Omega_{\boldsymbol{v}}\left[f(\boldsymbol{x}, \boldsymbol{v}', t)\right] \tag{2.13}$$

ここで，Sh はストローハル数，ϵ はクヌーセン数，$\Omega_{\boldsymbol{v}}$ は仮想粒子の衝突による増減を表す衝突演算子である．ただし，$\boldsymbol{v}'$ はパラメータ化されておりすべての $\boldsymbol{v}'$ について演算を行うと約束する．続いて，空間と時間を $\Delta x = 1/\epsilon$, $\Delta t = \mathrm{Sh}\Delta x$ によってそれぞれ分割し，式 (2.13) を時空間について一次前進差分によって近似すると次式を得る．

$$f(\boldsymbol{x} + \boldsymbol{v}\Delta t, \boldsymbol{v}, t + \Delta t) - f(\boldsymbol{x}, \boldsymbol{v}, t) = \Omega_{\boldsymbol{v}}\left[f(\boldsymbol{x}, \boldsymbol{v}', t)\right] \tag{2.14}$$

また, 離散化された時空間上での分布関数から, 流体の巨視的な密度 $\rho(\boldsymbol{x}, t)$ と巨視的な流速 $\boldsymbol{u}(\boldsymbol{x}, t)$ は以下のように定義される．

$$\rho(\boldsymbol{x}, t) = \sum_{\boldsymbol{v}} f(\boldsymbol{x}, \boldsymbol{v}, t) \tag{2.15}$$

$$\rho(\boldsymbol{x}, t)\boldsymbol{u}(\boldsymbol{x}, t) = \sum_{\boldsymbol{v}} \boldsymbol{v} f(\boldsymbol{x}, \boldsymbol{v}, t) \tag{2.16}$$

式 (2.14) において，左辺が並進，右辺が衝突を表していることに注意されたい．この右辺に次式で表される BGK(Bhatnagar-Gross-Krook) モデルと呼ばれる衝突演算子 [18] を導入する．

$$\Omega_{\boldsymbol{v}}\left[f(\boldsymbol{x}, \boldsymbol{v}', t)\right] = -\frac{1}{\tau}\left[f(\boldsymbol{x}, \boldsymbol{v}, t) - f^{eq}(\boldsymbol{x}, \boldsymbol{v}, t)\right] \tag{2.17}$$

ここで，τ は流体の無次元緩和時間，$f^{eq}(\boldsymbol{x}, \boldsymbol{v}, t)$ は局所平衡分布関数と呼ばれる関数である．式 (2.17) を式 (2.14) に代入すると，次式を得る．

$$f(\boldsymbol{x} + \boldsymbol{v}\Delta t, \boldsymbol{v}, t + \Delta t) = f(\boldsymbol{x}, \boldsymbol{v}, t) - \frac{1}{\tau}\left[f(\boldsymbol{x}, \boldsymbol{v}, t) - f^{eq}(\boldsymbol{x}, \boldsymbol{v}, t)\right] \tag{2.18}$$

最後に，局所平衡分布関数を求める．この関数は

$$\rho(\boldsymbol{x}, t) = \sum_{\boldsymbol{v}} f^{eq}(\boldsymbol{x}, \boldsymbol{v}, t) \tag{2.19}$$

$$\rho(\boldsymbol{x}, t)\boldsymbol{u}(\boldsymbol{x}, t) = \sum_{\boldsymbol{v}} \boldsymbol{v} f^{eq}(\boldsymbol{x}, \boldsymbol{v}, t) \tag{2.20}$$

を満たす必要がある．これらを満たす局所平衡分布関数はマクスウェル分布 [19] において $|\boldsymbol{u}|$ が二次の項まで展開すると得られて，以下のように定義される．

$$f^{eq}(\boldsymbol{x}, \boldsymbol{v}, t) = c(\boldsymbol{v})\rho(\boldsymbol{x}, t)\left[1 + 3\boldsymbol{v}\cdot\boldsymbol{u}(\boldsymbol{x}, t) + \frac{9}{2}(\boldsymbol{v}\cdot\boldsymbol{u}(\boldsymbol{x}, t))^2 - \frac{3}{2}\boldsymbol{u}(\boldsymbol{x}, t)^2\right] \tag{2.21}$$

ただし，式 (2.19), (2.20), (2.21) 内の $\rho(\boldsymbol{x}, t)$, $\boldsymbol{u}(\boldsymbol{x}, t)$ はそれぞれ式 (2.15), (2.16) で求めた値を用いる．また，$c(\boldsymbol{v})$ は速度ベクトル $\boldsymbol{v}$ に対する重みであり，D2Q9 モデルでは

$$c(\boldsymbol{v}) = \begin{cases} 4/9 & (\boldsymbol{v} = (0, 0)) \\ 1/9 & (\boldsymbol{v} = (\pm 1, 0), (0, \pm 1)) \\ 1/36 & (\boldsymbol{v} = (\pm 1, \pm 1)) \end{cases} \tag{2.22}$$

と求められる．以上の式 (2.18), (2.21), (2.15), 及び (2.16) により LBM の基本的な計算スキームが構成された．これらの計算スキームは Chapman-Enskog 展開によってナビエ・ストークス方程式を空間二次精度で再現することが知られている [20].

本研究においては時空間間隔をリスケールし，$\Delta x = 1$, $\Delta t = 1$ として計算を行った．その上で式 (2.18) は次のように並進と衝突の式に分離することができる．

$$f'(\boldsymbol{x} + \boldsymbol{v}, \boldsymbol{v}, t) = f(\boldsymbol{x}, \boldsymbol{v}, t) \tag{2.23}$$

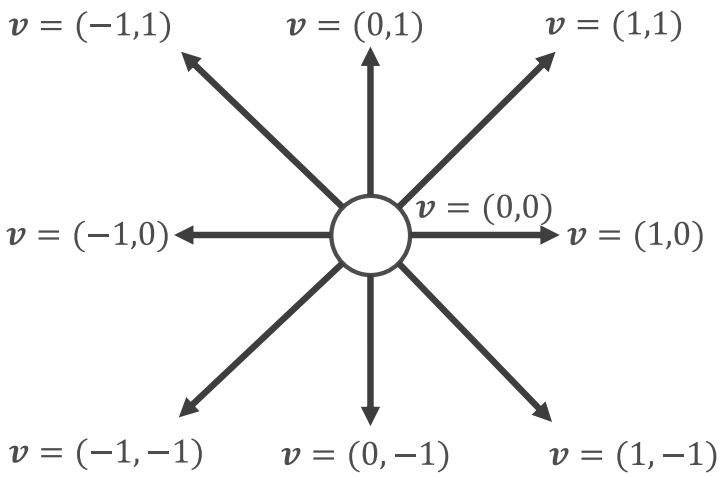

図 2.2　D2Q9 モデルの速度ベクトル

$$f(\boldsymbol{x}, \boldsymbol{v}, t+1) = f'(\boldsymbol{x}, \boldsymbol{v}, t) - \frac{1}{\tau}\left[f'(\boldsymbol{x}, \boldsymbol{v}, t) - f^{eq}(\boldsymbol{x}, \boldsymbol{v}, t)\right] \tag{2.24}$$

このとき，式 (2.21) は次のように表される．

$$f^{eq}(\boldsymbol{x}, \boldsymbol{v}, t) = c_i(\boldsymbol{v})\rho'(\boldsymbol{x}, t)\left[1 + 3\boldsymbol{v}\cdot\boldsymbol{u}'(\boldsymbol{x}, t) + \frac{9}{2}(\boldsymbol{v}\cdot\boldsymbol{u}'(\boldsymbol{x}, t))^2 - \frac{3}{2}\boldsymbol{u}'(\boldsymbol{x}, t)^2\right] \tag{2.25}$$

第 3 章

格子ボルツマン法を組み込んだ深層学習モデル

この章では，提案モデルの概要とより具体的な原理について説明する．

3.1　提案モデルの概要

提案モデルの概要図を図 3.1 に示す．この図が示すように，提案モデルは LBM をベースとしている．その並進と衝突の計算に重みを導入し，予測速度と圧力に対する実測速度と圧力との誤差からその重みを誤差逆伝播法によって学習することで，座標に依存する流体の振る舞いを精度良くシミュレーションすることを試みる．学習の流れを詳述する．まずある無次元時刻 t_0 の各格子点上での流体の速度, 圧力から, 仮想粒子の分布 $f(\boldsymbol{x}, \boldsymbol{v}, t_0)$ を導出する．そこから ΔT ステップに亘り，並進と衝突の計算によって仮想粒子の分布 $f'(\boldsymbol{x}, \boldsymbol{v}, t_0 + i)$, $f(\boldsymbol{x}, \boldsymbol{v}, t_0 + i + 1)$　$(i = 0, ..., \Delta T - 1)$ を逐次的に得る．最終的な時刻 $t_1 = t_0 + \Delta T$ での分布 $f(\boldsymbol{x}, \boldsymbol{v}, t_1)$ から巨視的な流体の速度と圧力を算出し，これと実測流体の速度と圧力との誤差から，各ステップに導入した重みを誤差逆伝播法によって学習する．

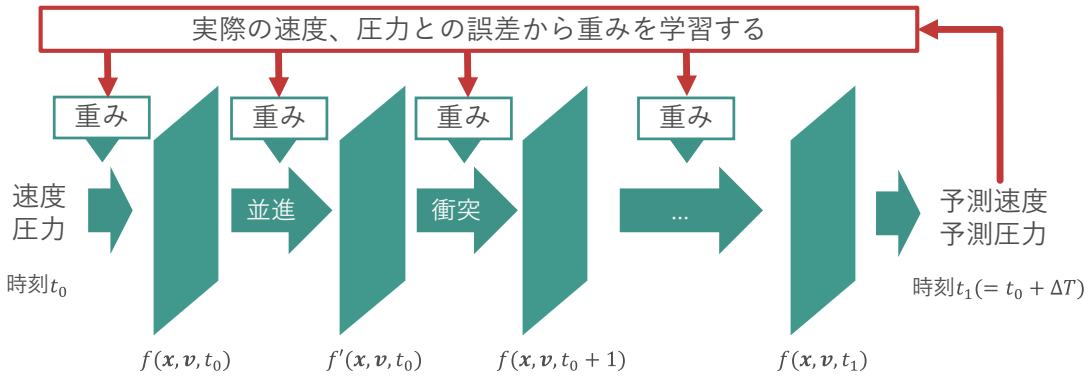

図 3.1　提案モデルの概要図

3.2　提案モデルの原理

この節では，具体的にどのように LBM に重みを導入し学習を行うか，その原理について説明する．まずは図 3.2(a) で表されるように，ある一時刻のデータのみが入力されたときにその ΔT 後の流速のみを予想する時系列性のないモデルを説明する．その後，図 3.2(b) で表されるように ΔT おきの時系列データが入力されたとき，それぞれの時刻に対して ΔT 後の時系列データを予測するモデルを説明する．

3.2.1　時系列性のないモデル

時系列性のないモデルでは，ある無次元時刻 t_0 での流速 $\hat{\boldsymbol{u}}(\boldsymbol{x}, t_0)$ と圧力 $\hat{\rho}(\boldsymbol{x}, t_0)$ のデータが入力されたときに時刻 $t_1 = t_0 + \Delta T$ での流速 $\hat{\boldsymbol{u}}(\boldsymbol{x}, t_1)$ と圧力 $\hat{\rho}(\boldsymbol{x}, t_1)$ を予測する．まず LBM の並進と衝突の式にどのように重みを導入するかを説明する．

LBM の並進は，仮想粒子の分布関数 $f(\boldsymbol{x}, \boldsymbol{v}, t)$ を用いて以下のように表された (再掲)．

$$f'(\boldsymbol{x} + \boldsymbol{v}, \boldsymbol{v}, t) = f(\boldsymbol{x}, \boldsymbol{v}, t) \tag{2.23}$$

この式に重み $w'_{\mathrm{const}}(\boldsymbol{x}, \boldsymbol{v}, t)$, $w'_{\mathrm{fprev}}(\boldsymbol{x}, \boldsymbol{v}, t)$ を導入する．重みを導入することで，仮想粒子の分布関数 $f(\boldsymbol{x}, \boldsymbol{v}, t)$ を並進させる際に，仮想粒子の流入や流出を表現することができる．重みを導入した並進の式は以下のように表される．

$$f'(\boldsymbol{x} + \boldsymbol{v}, \boldsymbol{v}, t) = w'_{\mathrm{const}}(\boldsymbol{x}, \boldsymbol{v}, t) + w'_{\mathrm{fprev}}(\boldsymbol{x}, \boldsymbol{v}, t) f(\boldsymbol{x}, \boldsymbol{v}, t) \tag{3.1}$$

また，衝突の式は以下のように表された (再掲)．

$$f(\boldsymbol{x}, \boldsymbol{v}, t + 1) = f'(\boldsymbol{x}, \boldsymbol{v}, t) - \frac{1}{\tau} \left[f'(\boldsymbol{x}, \boldsymbol{v}, t) - f^{eq}(\boldsymbol{x}, \boldsymbol{v}, t) \right] \tag{2.24}$$

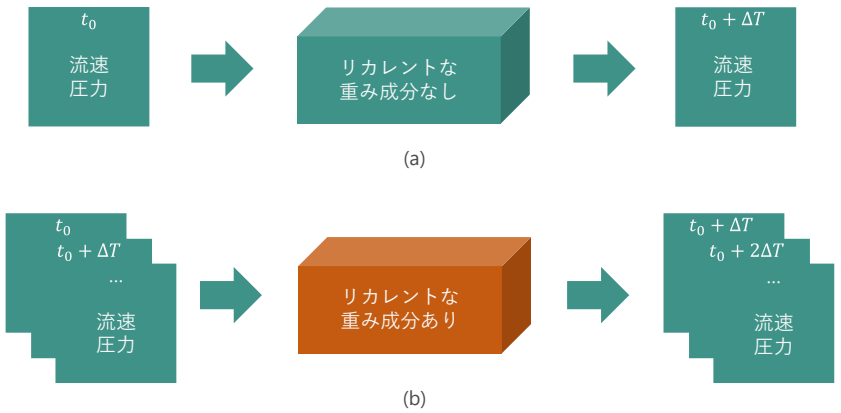

図 3.2　時系列性のないモデルと時系列性のあるモデルの比較 (a) 時系列性のないモデルの入出力 (b) 時系列性のあるモデルの入出力

$$f^{eq}(\boldsymbol{x},\boldsymbol{v},t) = c_i(\boldsymbol{v})\rho'(\boldsymbol{x},t)\left[1+3\boldsymbol{v}\cdot\boldsymbol{u}'(\boldsymbol{x},t)+\frac{9}{2}(\boldsymbol{v}\cdot\boldsymbol{u}'(\boldsymbol{x},t))^2-\frac{3}{2}\boldsymbol{u}'(\boldsymbol{x},t)^2\right] \tag{2.25}$$

これらの式にも重み $w_{\mathrm{fprev}}(\boldsymbol{x},\boldsymbol{v},t)$, $w_{\mathrm{vu}}(\boldsymbol{x},\boldsymbol{v},t)$, $w_{\mathrm{vxu}}(\boldsymbol{x},\boldsymbol{v},t)$, $w_{\mathrm{vu2}}(\boldsymbol{x},\boldsymbol{v},t)$, $w_{\mathrm{u2}}(\boldsymbol{x},\boldsymbol{v},t)$ を導入する．重みを導入した衝突の式はそれぞれ以下のように表される．

$$f(\boldsymbol{x},\boldsymbol{v},t+1) = w_{\mathrm{fprev}}(\boldsymbol{x},\boldsymbol{v},t)f'(\boldsymbol{x},\boldsymbol{v},t)+(1-w_{\mathrm{fprev}}(\boldsymbol{x},\boldsymbol{v},t))\,f^{eq}(\boldsymbol{x},\boldsymbol{v},t) \tag{3.2}$$

$$\begin{aligned} f^{eq}(\boldsymbol{x},\boldsymbol{v},t) =& c_i(\boldsymbol{v})\rho'(\boldsymbol{x},t) \\ & [1+w_{\mathrm{vu}}(\boldsymbol{x},\boldsymbol{v},t)\boldsymbol{v}\cdot\boldsymbol{u}'(\boldsymbol{x},t) \\ & +w_{\mathrm{vxu}}(\boldsymbol{x},\boldsymbol{v},t)\boldsymbol{v}\times\boldsymbol{u}'(\boldsymbol{x},t) \\ & +w_{\mathrm{vu2}}(\boldsymbol{x},\boldsymbol{v},t)(\boldsymbol{v}\cdot\boldsymbol{u}'(\boldsymbol{x},t))^2 \\ & +w_{\mathrm{u2}}(\boldsymbol{x},\boldsymbol{v},t)\boldsymbol{u}'(\boldsymbol{x},t)^2] \end{aligned} \tag{3.3}$$

ただし，$\times$ はベクトル $\boldsymbol{u}=(u_1,u_2)$, $\boldsymbol{v}=(v_1,v_2)$ に対して

$$\boldsymbol{u}\times\boldsymbol{v} = u_1v_2-u_2v_1 \tag{3.4}$$

と定義される演算子である．ここで, 式 (2.25) と比べて式 (3.3) には新たに $w_{\mathrm{vxu}}(\boldsymbol{x},\boldsymbol{v},t)\boldsymbol{v}\times\boldsymbol{u}(\boldsymbol{x},t)$ という項が導入されている．これは，座標 $\boldsymbol{x}$ における流体の回転を表現するためである．

次に，どのように流速と圧力のデータを入力層へと入力するか，出力層から流速と圧力のデータを取得して損失を計算するか説明する．与えられた時刻 t_0 での流速と圧力のデータだけでは仮想粒子の分布状態 $f(\boldsymbol{x},\boldsymbol{v},t_0)$ を計算することはできないが，ここでは局所平衡分布関数に重みを導入した式 (3.3) を用いて次式のように入力層の分布状態を推定した．

$$\begin{aligned} f(\boldsymbol{x},\boldsymbol{v},t_0) =& c_i(\boldsymbol{v})\hat{\rho}(\boldsymbol{x},t_0) \\ & [1+w_{\mathrm{vu}}(\boldsymbol{x},\boldsymbol{v},t_0)\boldsymbol{v}\cdot\hat{\boldsymbol{u}}(\boldsymbol{x},t_0) \\ & +w_{\mathrm{vxu}}(\boldsymbol{x},\boldsymbol{v},t_0)\boldsymbol{v}\times\hat{\boldsymbol{u}}(\boldsymbol{x},t_0) \\ & +w_{\mathrm{vu2}}(\boldsymbol{x},\boldsymbol{v},t_0)(\boldsymbol{v}\cdot\hat{\boldsymbol{u}}(\boldsymbol{x},t_0))^2 \\ & +w_{\mathrm{u2}}(\boldsymbol{x},\boldsymbol{v},t_0)\hat{\boldsymbol{u}}(\boldsymbol{x},t_0)^2] \end{aligned} \tag{3.5}$$

また，出力層では単純に仮想粒子の分布 $f(\boldsymbol{x},\boldsymbol{v},t_1)$ から式 (2.16), (2.15) を用いて流速 $\boldsymbol{u}(\boldsymbol{x},t_1)$ と圧力 $\rho(\boldsymbol{x},t_1)$ を算出した．これらと実測流速 $\hat{\boldsymbol{u}}(\boldsymbol{x},t_1)$ と圧力 $\hat{\rho}(\boldsymbol{x},t_1)$ から求められる二乗損失は，式 (2.4) から次のように表される．ただし，ここでは係数を 1 とした．

$$E=\sum_{\boldsymbol{x}}\|\boldsymbol{u}(\boldsymbol{x},t_1)-\hat{\boldsymbol{u}}(\boldsymbol{x},t_1)\|^2+|\rho(\boldsymbol{x},t_1)-\hat{\rho}(\boldsymbol{x},t_1)|^2 \tag{3.6}$$

以上の重みの初期値の設定として，通常の LBM と同じ係数となるように以下のように定めた．

$$\begin{cases} w'_{\mathrm{const}}(\boldsymbol{x},\boldsymbol{v},t) = 0 \\ w'_{\mathrm{fprev}}(\boldsymbol{x},\boldsymbol{v},t) = 1 \\ w_{\mathrm{fprev}}(\boldsymbol{x},\boldsymbol{v},t) = 1 - 1/\tau \\ w_{\mathrm{vu}}(\boldsymbol{x},\boldsymbol{v},t) = 3 \\ w_{\mathrm{vxu}}(\boldsymbol{x},\boldsymbol{v},t) = 0 \\ w_{\mathrm{vu2}}(\boldsymbol{x},\boldsymbol{v},t) = 9/2 \\ w_{\mathrm{u2}}(\boldsymbol{x},\boldsymbol{v},t) = -3/2 \end{cases} \tag{3.7}$$

この初期値から学習を進めることで重みを更新していく．例として, 中間層の重み $w'_{\mathrm{fprev}}(\boldsymbol{x},\boldsymbol{v},t)$ の更新式を示す．最も単純な勾配降下法では，式 (2.5), (2.9), (3.1), (3.6) より以下のように表される．

$$w'_{\mathrm{fprev}}(\boldsymbol{x},\boldsymbol{v},t) \leftarrow w'_{\mathrm{fprev}}(\boldsymbol{x},\boldsymbol{v},t) - \eta\delta'(\boldsymbol{x}+\boldsymbol{v},\boldsymbol{v},t)f(\boldsymbol{x},\boldsymbol{v},t) \tag{3.8}$$

ただし，$\delta'(\boldsymbol{x}+\boldsymbol{v},\boldsymbol{v},t)$ は $f'(\boldsymbol{x}+\boldsymbol{v},\boldsymbol{v},t)$ の誤差を表している (式 2.11 を参照)．実行する上では pyTorch によって自動微分が行われるので，このような更新式の導出は必要ない．

注意すべき点として，式 (3.1) からわかるように並進をするとき最も外側の座標の仮想粒子の分布関数は算出することができない．そのため，並進をするたびに外枠 1 マス分の分布関数は消滅してしまうので，入力されるデータの大きさに比べて，出力されるデータの大きさは並進の回数分小さくなる．図 3.3 に並進時の外枠 1 マス分に関する分布関数の様子を示す．この図において，各矢印は $\boldsymbol{v}$ に対応しており (各格子点上の中央の丸は $\boldsymbol{v}=\boldsymbol{0}$ に対応する)，実線は並進を行える仮想粒子の速度成分を表し，破線は並進を行うことのできない速度成分を表している．

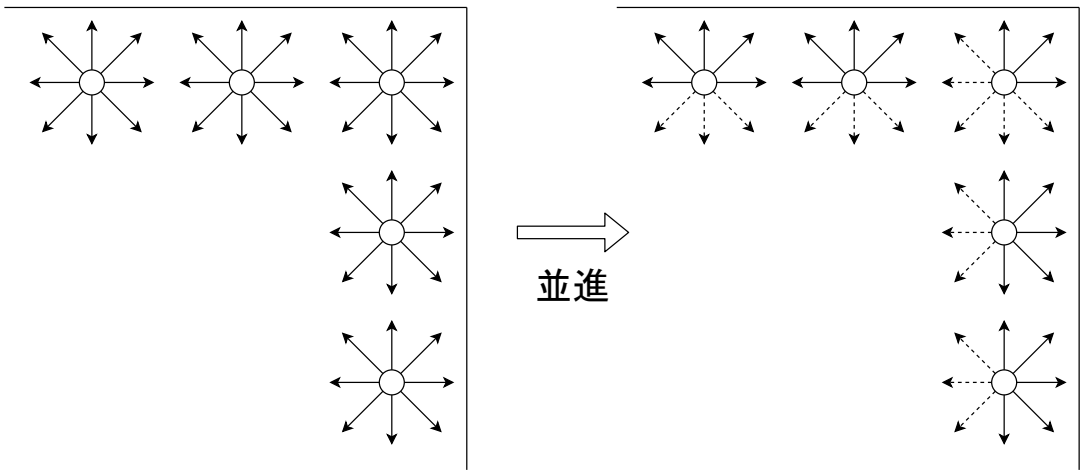

図 3.3　外枠 1 マス分の仮想粒子の分布関数

3.2.2 時系列性のあるモデル

実験で用いたモデルには入力データの時系列性を反映させるため，更にリカレントな重みを導入した．図 3.4 では本モデルの概要を示す．このモデルでは，ΔT おきの時系列データが入力されたとき，それぞれの時刻に対して ΔT 後の時系列データを予測する．

まずこの図に書かれるように，記号を再定義する．時系列データの長さを n とする．そして無次元時刻 t_0 から t_n までの Δt おきの実測時系列データを

$$(\hat{\boldsymbol{u}}(\boldsymbol{x}, t_0), \hat{\rho}(\boldsymbol{x}, t_0)), (\hat{\boldsymbol{u}}(\boldsymbol{x}, t_1), \hat{\rho}(\boldsymbol{x}, t_1)), \cdots, (\hat{\boldsymbol{u}}(\boldsymbol{x}, t_n), \hat{\rho}(\boldsymbol{x}, t_n)) \tag{3.9}$$

ただし，

$$t_i = t_0 + i\Delta T \quad (0 \leq i \leq n) \tag{3.10}$$

のように表す．なお，$\hat{\boldsymbol{u}}(\boldsymbol{x}, t_i)$ は実測の無次元流速，$\hat{\rho}(\boldsymbol{x}, t_i)$ は実測の無次元圧力を表す $(0 \leq i \leq n)$．更に，時刻 t_i $(0 \leq i \leq n-1)$ のデータが入力されたときの内部における時刻 $t_i + t (0 \leq t \leq \Delta T)$ の衝突 (並進) 後の仮想粒子の分布関数とそれから式 (2.16), (2.15) によって得られる巨視的な流速，圧力をそれぞれ

$$f_i^{(\prime)}(\boldsymbol{x}, \boldsymbol{v}, t) \tag{3.11}$$

$$\boldsymbol{u}_i^{(\prime)}(\boldsymbol{x}, t) \tag{3.12}$$

$$\rho_i^{(\prime)}(\boldsymbol{x}, t) \tag{3.13}$$

と表すことにする．局所平衡分布関数 $f_i^{eq}(\boldsymbol{x}, \boldsymbol{v}, t)$ も同様である．

まずはこの時系列性があるモデルの入出力を概観する．このモデルのタスクは，$(\hat{\boldsymbol{u}}(\boldsymbol{x}, t_i), \hat{\rho}(\boldsymbol{u}, t_i))$ が $i = 0$ から $i = n-1$ まで順次与えられたときにそれぞれに対して $(\hat{\boldsymbol{u}}(\boldsymbol{x}, t_{i+1}), \hat{\rho}(\boldsymbol{x}, t_{i+1}))$ を予測することである．モデルの予測流速と圧力は，式 (3.12), (3.13) から $\boldsymbol{u}_i(\boldsymbol{x}, \Delta T)$，$\rho_i(\boldsymbol{x}, \Delta T)$ として得られ，式 (2.4) より二乗損失は次のように表される．

$$E = \sum_{\substack{\boldsymbol{x} \\ 0 \leq i \leq n-1}} \|\boldsymbol{u}_i(\boldsymbol{x}, \Delta T) - \hat{\boldsymbol{u}}(\boldsymbol{x}, t_{i+1})\|^2 + |\rho_i(\boldsymbol{x}, \Delta T) - \hat{\rho}(\boldsymbol{x}, t_{i+1})|^2 \tag{3.14}$$

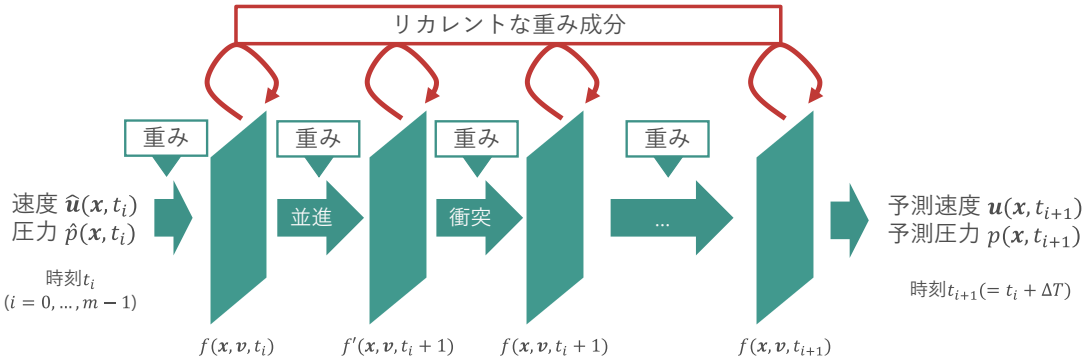

図 3.4　時系列性のあるモデルの概要図

さて，このモデルでは重みづけられた並進の式 (3.1) は更に重み成分 $w'_{\mathrm{recur}}(\boldsymbol{x},\boldsymbol{v},t)$ が加えられ次のように拡張される (ここで，各重みの変数 $(\boldsymbol{x},\boldsymbol{v},t)$ は省略してあることに注意せよ).

$$f'_i(\boldsymbol{x}+\boldsymbol{v},\boldsymbol{v},t) = \begin{cases} w'_{\mathrm{const}} + w'_{\mathrm{fprev}} f_i(\boldsymbol{x},\boldsymbol{v},t) & (i=0) \\ w'_{\mathrm{recur}} f_{i-1}(\boldsymbol{x}+\boldsymbol{v},\boldsymbol{v},t) & \\ \qquad + (1-w'_{\mathrm{recur}})\left(w'_{\mathrm{const}} + w'_{\mathrm{fprev}} f_i(\boldsymbol{x},\boldsymbol{v},t)\right) & (0<i\leq n-1) \end{cases} \tag{3.15}$$

続いて，重みづけられた衝突の式 (3.2), (3.3) には更に重み成分 $w_{\mathrm{recur}}(\boldsymbol{x},\boldsymbol{v},t)$ が加えられそれぞれ次のように拡張される (こちらも各重みの変数 $(\boldsymbol{x},\boldsymbol{v},t)$ を省略した).

$$f_i(\boldsymbol{x},\boldsymbol{v},t{+}1) = \begin{cases} w_{\mathrm{fprev}} f'_i(\boldsymbol{x},\boldsymbol{v},t) + (1-w_{\mathrm{fprev}}) f_i^{eq}(\boldsymbol{x},\boldsymbol{v},t) & (i=0) \\ w_{\mathrm{recur}} f_{i-1}(\boldsymbol{x},\boldsymbol{v},t+1) & \\ \quad + (1-w_{\mathrm{recur}})\left(w_{\mathrm{fprev}} f'_i(\boldsymbol{x},\boldsymbol{v},t) + (1-w_{\mathrm{fprev}}) f_i^{eq}(\boldsymbol{x},\boldsymbol{v},t)\right) & (0<i\leq n-1) \end{cases} \tag{3.16}$$

$$\begin{aligned} f_i^{eq}(\boldsymbol{x},\boldsymbol{v},t) =& c_i(\boldsymbol{v})\rho'_i(\boldsymbol{x},t) \\ & [1 + w_{\mathrm{vu}}\boldsymbol{v}\cdot\boldsymbol{u}'_i(\boldsymbol{x},t) \\ & + w_{\mathrm{vxu}}\boldsymbol{v}\times\boldsymbol{u}'_i(\boldsymbol{x},t) \\ & + w_{\mathrm{vu2}}(\boldsymbol{v}\cdot\boldsymbol{u}'_i(\boldsymbol{x},t))^2 \\ & + w_{\mathrm{u2}}\boldsymbol{u}'_i(\boldsymbol{x},t)^2] \end{aligned} \tag{3.17}$$

以上のようにしてリカレントな重み成分をもつ時系列性のあるモデルを構築した.

第 4 章

提案モデルによる日本近辺の風速予測

本論文では，提案モデルの性能検証のために日本近辺の風速予測を実施した．また，性能の比較のために CNN によるエンコーダ・デコーダと LSTM を用いた U-Net-like なモデルを実装し，それぞれのモデルの性能を比較した．

4.1 実験の概要

本実験では，提案モデルと 4.4.1 項で説明する U-Net-like なモデルの 2 つのモデルに 3 時刻分の時系列データを入力し，それぞれの 3 時間後の風速を予測するように訓練した．その学習と評価には，4.2 節で述べるように気象庁が提供している日本近辺の風速と気圧データを用いた．

モデルが出力した最終時刻の予測値と実測値を用いて，以下の評価指標を算出した．ここで，指標に用いられる「風速」はベクトル量ではなく，スカラー量であることに注意されたい．

1. 全体の風速の RMSE[m/s](Root Mean Squared Error, 二乗平均平方根誤差) と提案モデルの RMSE 改善率 [%]
2. 全体の風速の ME[m/s](Mean Error, 平均誤差)
3. 全体の風向の RMSE[°] と提案モデルの RMSE 改善率 [%]
4. 座標ごとの風速の RMSE[m/s] と提案モデルの RMSE 改善率 [%]
5. 座標ごとの風速の ME[m/s]
6. 座標ごとの風向の RMSE[°] と提案モデルの RMSE 改善率 [%]

これらの評価指標は一般的に気象予報モデルの統計的検証に用いられるものを参考にした [21]．以下に RMSE 改善率以外のそれぞれの評価指標の定義を示す．

$$(\text{全体の風速の RMSE[m/s]}) = \sqrt{\frac{1}{NM}\sum_{\boldsymbol{x},i}(|\boldsymbol{w}(\boldsymbol{x},t_i)| - |\hat{\boldsymbol{w}}(\boldsymbol{x},t_i)|)^2} \tag{4.1}$$

$$(\text{全体の風速の ME[m/s]}) = \frac{1}{NM}\sum_{\boldsymbol{x},i}(|\boldsymbol{w}(\boldsymbol{x},t_i)| - |\hat{\boldsymbol{w}}(\boldsymbol{x},t_i)|) \tag{4.2}$$

$$(\text{全体の風向の RMSE}[^\circ]) = \sqrt{\frac{1}{NM}\sum_{\boldsymbol{x},i} \arg(\boldsymbol{w}(\boldsymbol{x},t_i), \hat{\boldsymbol{w}}(\boldsymbol{x},t_i))^2} \tag{4.3}$$

$$(\text{座標ごとの風速の RMSE}[\text{m/s}]) = \sqrt{\frac{1}{N}\sum_{\boldsymbol{i}} (|\boldsymbol{w}(\boldsymbol{x},t_i)| - |\hat{\boldsymbol{w}}(\boldsymbol{x},t_i)|)^2} \tag{4.4}$$

$$(\text{座標ごとの風速の ME}[\text{m/s}]) = \frac{1}{N}\sum_{\boldsymbol{i}} (|\boldsymbol{w}(\boldsymbol{x},t_i)| - |\hat{\boldsymbol{w}}(\boldsymbol{x},t_i)|) \tag{4.5}$$

$$(\text{座標ごとの風向の RMSE}[^\circ]) = \sqrt{\frac{1}{N}\sum_{\boldsymbol{i}} \arg(\boldsymbol{w}(\boldsymbol{x},t_i), \hat{\boldsymbol{w}}(\boldsymbol{x},t_i))^2} \tag{4.6}$$

ただし，M は格子点 $\boldsymbol{x}$ の総数，N はデータセットの総数とし，そのうちの i 番目のデータセットについて $\boldsymbol{w}(\boldsymbol{x},t_i)$ をモデルによる予測風速ベクトル，$\hat{\boldsymbol{w}}(\boldsymbol{x},t_i)$ を実測の風速ベクトルとした．ここで，$\boldsymbol{w}(\boldsymbol{x},t_i)$ と $\hat{\boldsymbol{w}}(\boldsymbol{x},t_i)$ は共に速度の次元を持つ．また，arg は二つのベクトルが成す角度を返す関数である．

また，RMSE 改善率は以下の式で定義される．

$$(\text{RMSE 改善率 }[\%]) = \frac{\text{RMSE}_{\text{u-net}} - \text{RMSE}_{\text{prop}}}{\text{RMSE}_{\text{u-net}}} \times 100 \tag{4.7}$$

ただし, $\text{RMSE}_{\text{u-net}}$ は U-Net-like なモデルの RMSE, $\text{RMSE}_{\text{prop}}$ は提案モデルの RMSE を表す.

4.2 利用したデータ及び学習条件

4.2.1 利用したデータ

気象庁が提供している日本近辺の風速と気圧データを京都大学の生存圏データベース [22] から入手した．このデータセットは図 4.1 に示す通り，日本近辺の北緯 22.4° から 47.6° まで，東経 120° から 150° までの領域をそれぞれ 0.05° × 0.0625° の細かさで等緯度等経度に区切り，各格子点上における風速と気圧を記録したものである [23]．すなわち，ある一時刻のデータは緯度経度について 505 × 480 の行列となっており，ある一点には風速の南北方向成分 [m/s], 風速の東西方向成分 [m/s], 気圧 [hPa] の 3 つの値が記録されている．

提供されているデータは 3 時間間隔であり，(前日の)21:00, 0:00, 3:00, 6:00 の 4 時刻分をひとまとめにして時系列データとした．これを 2011 年 1 月 1 日から 2020 年 12 月 31 日までの 3650 日分用意し (統計を取る際の簡便化のために閏日は除いてある)，4.2.2 項で述べる処理をする前の全体のデータセットとした．

4.2.2 データの前処理

4.2.1 項で述べたデータセットをそのまま入出力に用いるのではなく，前処理を実施した．その処理の詳細を以下に示す．

図 4.2 に示すように，505 × 480 の行列を更に 10 × 10 の格子によって区切り，この格子内で風速と気圧を平均化することで 50 × 48 のサイズまで落とした．なお，端数分は切り捨てている．これは，提案モデルの格子の大きさを適切に設定することで，3 時間で変化する風速の空間的な変化を捉えることができると考えたためである．このデータセットを改めて全体のデータセットとした．

4.2.3　学習条件

続いて，モデルの詳細な学習条件について述べる．提案モデルのアーキテクチャについての概要を 3.2.2 項で述べたがここではより具体的に数値を提示し，モデルの仕様を詳細化する．図 4.3 に示したように，このモデルでは並進と衝突をそれぞれ 5 回行った (緑の実線矢印が並進を表し，緑の破線矢印が衝突を表している)．すなわち式 (3.10) において $\Delta T = 5$ である．3.2.1 項で述べた通り衝突の回数分だけ外枠が削られるため，図中の赤破線によって強調されているように出力層の大きさは入力層に比べて東西南北の端 5 マス分減少している．(前日の)21:00, 0:00, 3:00 の 3

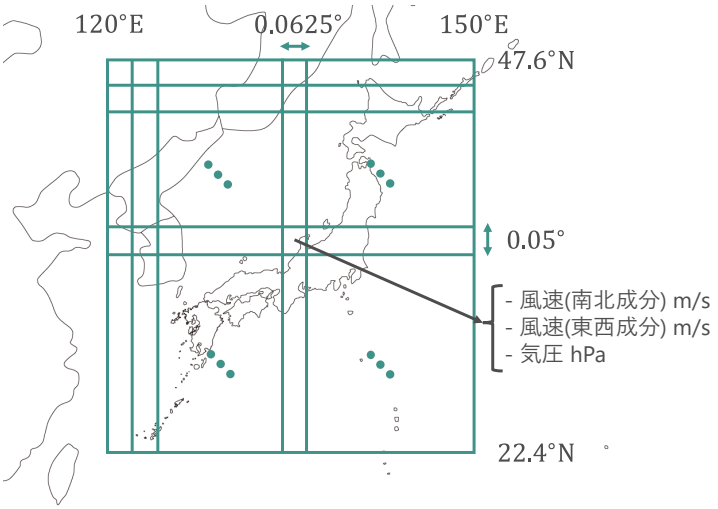

図 4.1　気象庁が提供している日本近辺の風速と気圧データの領域

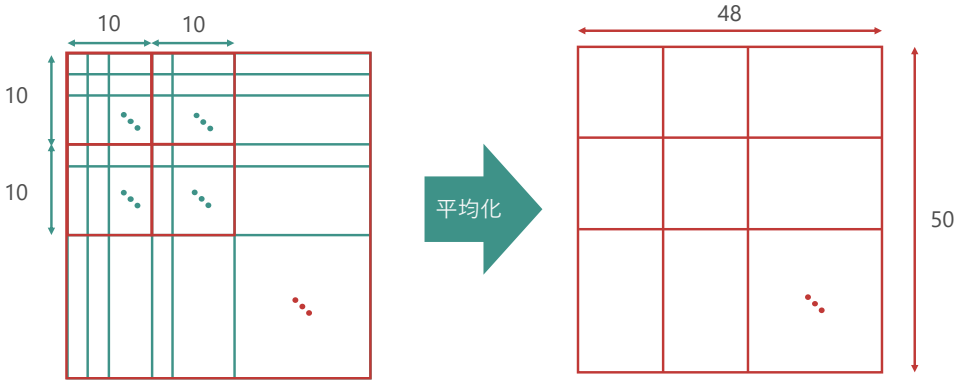

図 4.2　風速と気圧の平均化

時刻分のデータを入力として 0:00, 3:00, 6:00 の風速を予測させた．すなわち，式 (3.9) において $n = 4$，式 (3.10) において無次元時刻 ΔT は 3[h] 相当である．

提案モデルの学習には Adam[24] を用いた．学習率は 10^{-3} とし，ミニバッチサイズは 16 とした．全体のデータセットを 2920 日分と 730 日分に分けそれぞれを学習用データと検証用データとした．学習は 500 エポック行い，かかった時間は約 8 時間であった．実行環境には Google Colaboratory[25] を用い，GPU は Tesla T4 を用いた．また提案モデルの構築には PyTorch[15] を用い，自動微分による学習を行った．

4.3　実験結果

4.1 節で述べた評価指標を用いて，提案モデルの性能を評価した．その結果を順に述べる．提案モデルの全体の風速の RMSE[m/s], ME[m/s] はそれぞれ 1.0432[m/s], −0.1909[m/s] であった．提案モデルの全体の風向の RMSE[°] は 21.444[°] であった．また，提案モデルの座標ごとの風速の RMSE[m/s], ME[m/s] と風向の RMSE[°] はそれぞれ図 4.5(a), 4.6(a), 4.7(a) に示す通りであった．

4.4　物理的な構造を含まない深層学習モデルによる実験結果

4.4.1　物理的な構造を含まない深層学習モデル

提案モデルと比較するために用いた物理的な構造をもたない U-Net-like なモデルを図 4.4 に示す．このモデルは 1.2.1 項で述べた Chen らのモデルと U-Net[26] を参考にしたものである．図中の各層の左に大きさを，上部にチャンネル数を記載してある．入出力部分では，風速の東西方向成分と南北方向成分，そして気圧を正規化して 3 チャンネルにまとめた．エンコーダ・デコーダの各層には畳み込み層とバッチ正規化 [27]，活性化関数である tanh を用いた．また，エンコーダの各層の出力をコピーしてデコーダの対応する層の入力に結合するスキップコネクションを用いた．エ

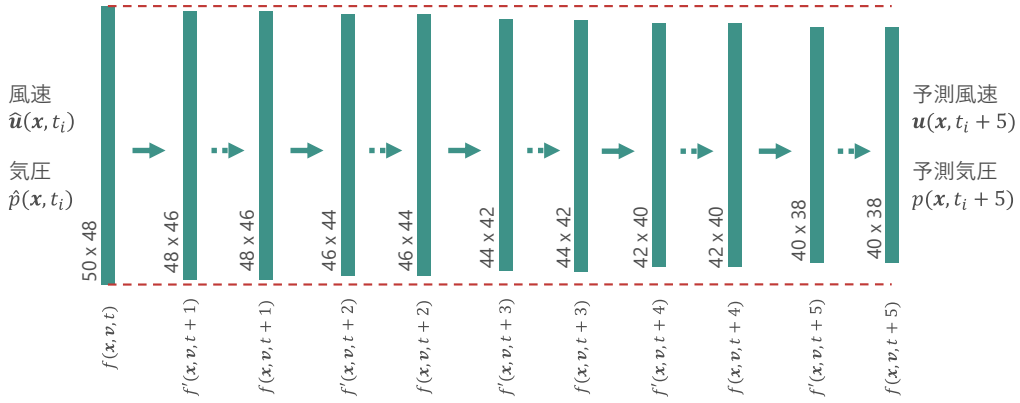

図 4.3　モデルのアーキテクチャ

ンコーダから出力される信号を平滑化してから LSTM ユニットに入力し，その出力をデコーダに入力した．デコードの際にはサイズを復元するために逆畳み込み層を用いた．出力の際に，提案モデルと性能比較をできるように周囲 5 マス分を切り捨てた．

その他の利用したデータ及び学習条件は 4.2 節と同様である．

4.4.2 実験結果

4.1 節で述べた評価指標を用いて，U-Net-like なモデルの性能を評価し，提案モデルと比較した．U-Net-like なモデルについて，全体の風速の RMSE[m/s] と ME[m/s]，そして風向の RMSE[°] をそれぞれ提案モデルの結果と比較しながら，それぞれ表 4.1 に示す．また，座標ごとの風速の RMSE[m/s] と ME[m/s]，そして風向の RMSE[°] をそれぞれ提案モデルの結果と比較しながら，それぞれ図 4.5, 4.6, 4.7 に示す．

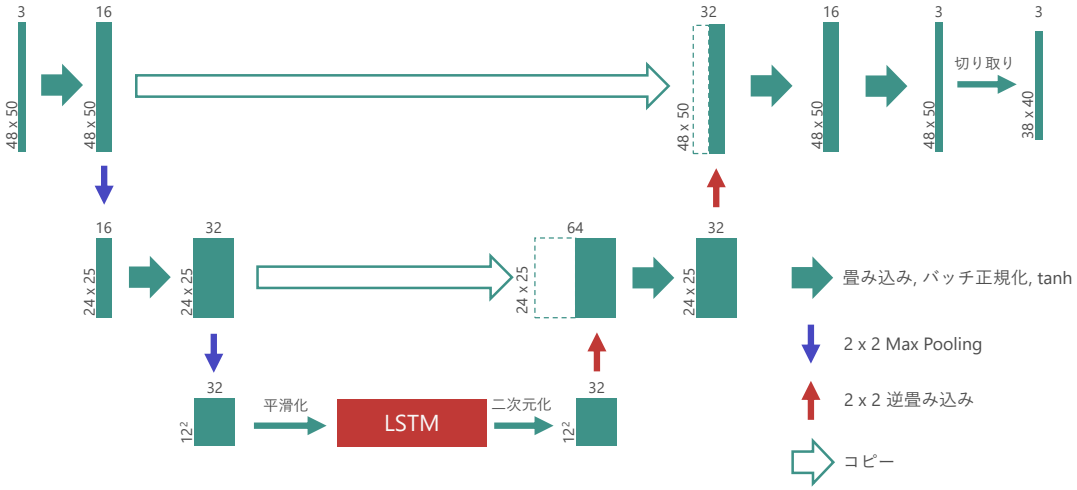

図 4.4　物理的な構造を含まない深層学習モデルのアーキテクチャ

表 4.1　全体の風速の RMSE[m/s]

提案モデル	U-Net-like なモデル	RMSE 改善率
1.0432	1.0930	4.57

表 4.2　全体の風速の ME[m/s]

提案モデル	U-Net-like なモデル
-0.1909	0.2001

表 4.3　全体の風向の RMSE[°]

提案モデル	U-Net-like なモデル	RMSE 改善率
21.444	22.521	4.78

4.5　考察

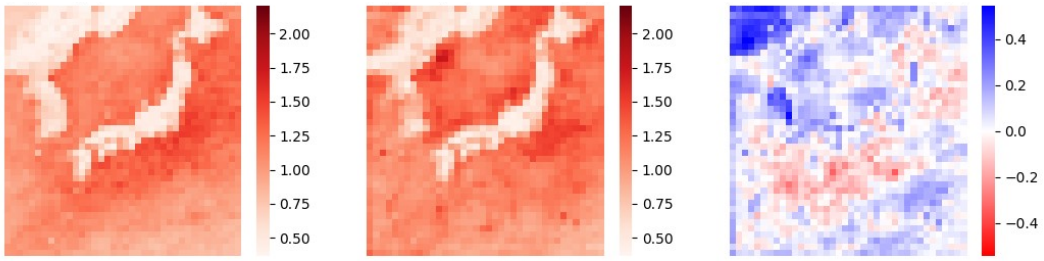

図 4.5　座標ごとの風速の RMSE[m/s] (a) 提案モデルの RMSE[m/s] (b)U-Net-like モデルの RMSE[m/s] (c) 提案モデルによる RMSE 改善率 [%]

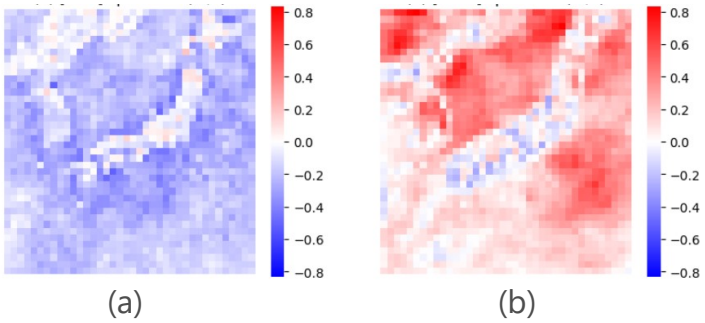

図 4.6　座標ごとの風速の ME[m/s] (a) 提案モデルの ME[m/s] (b)U-Net-like モデルの ME[m/s]

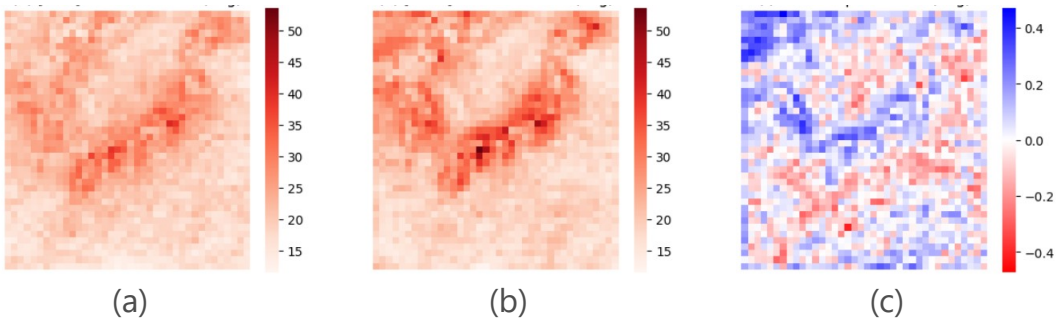

図 4.7　座標ごとの風向の RMSE[°] (a) 提案モデルの RMSE[°] (b)U-Net-like モデルの RMSE[°] (c) 提案モデルによる RMSE 改善率 [%]

参考文献

[1] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, Vol. 313, No. 5786, pp. 504–507, 2006.

[2] Martin Schultz, Clara Betancourt, Bing Gong, Felix Kleinert, Michael Langguth, Lukas Leufen, Amirpasha Mozaffari, and Scarlet Stadtler. Can deep learning beat numerical weather prediction? *Philosophical Transactions of The Royal Society A Mathematical Physical and Engineering Sciences*, Vol. 379, p. 20200097, 02 2021.

[3] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, Vol. 9, No. 4, pp. 611–629, Aug 2018.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, Vol. 9, No. 8, pp. 1735–1780, nov 1997.

[5] Yaoran Chen, Yan Wang, Zhikun Dong, Jie Su, Zhaolong Han, Dai Zhou, Yongsheng Zhao, and Yan Bao. 2-d regional short-term wind speed forecast based on cnn-lstm deep learning model. *Energy Conversion and Management*, Vol. 244, p. 114451, 2021.

[6] Shiyi Chen and Gary D. Doolen. Lattice boltzmann method for fluid flows. *Annual Review of Fluid Mechanics*, Vol. 30, No. 1, pp. 329–364, 1998.

[7] N. Satofuka and T. Nishioka. Parallelization of lattice boltzmann method for incompressible flow computations. *Computational Mechanics*, Vol. 23, No. 2, pp. 164–171, Mar 1999.

[8] Kyoung-Su Oh and Keechul Jung. Gpu implementation of neural networks. *Pattern Recognition*, Vol. 37, No. 6, pp. 1311–1314, 2004.

[9] A.K. Jain, Jianchang Mao, and K.M. Mohiuddin. Artificial neural networks: a tutorial. *Computer*, Vol. 29, No. 3, pp. 31–44, 1996.

[10] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, Vol. 367, No. 6481, pp. 1026–1030, 2020.

[11] Joseph Pateras, Pratip Rana, and Preetam Ghosh. A taxonomic survey of physics-informed machine learning. *Applied Sciences*, Vol. 13, No. 12, 2023.

[12] Marcel Lesieur. *Diffusion of Passive Scalars*, pp. 205–241. Springer Netherlands, Dordrecht, 1990.

[13] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert L. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, Vol. 2, pp. 359–366, 1989.

[14] Atılım Günes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: A survey. *J. Mach. Learn. Res.*, Vol. 18, No. 1, pp. 5595–5637, jan 2017.

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

[16] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[17] Takaji Inamuro and Bradford Sturtevant. Numerical study of discrete-velocity gases. *Physics of Fluids A: Fluid Dynamics*, Vol. 2, No. 12, pp. 2196–2203, 12 1990.

[18] P. L. Bhatnagar, E. P. Gross, and M. Krook. A model for collision processes in gases. i. small amplitude processes in charged and neutral one-component systems. *Phys. Rev.*, Vol. 94, pp. 511–525, May 1954.

[19] 日本流体力学会編. 流体力学シリーズ. 朝倉書店, 1989.7.

[20] 稲室隆二. 格子ボルツマン法入門 : 複雑境界および移動境界流れの数値計算法. 丸善出版, 2020.1.

[21] 気象庁. 付録 d 数値予報研修テキストで用いた表記と統計的検証に用いる代表的な指標, 2018. https://www.jma.go.jp/jma/kishou/books/nwptext/52/Appendix_D.pdf, (参照 2024-01-16).

[22] 京都大学生存圏研究所. 生存圏データベース., 2004. http://database.rish.kyoto-u.ac.jp/, (参照 2024-01-16).

[23] 一般財団法人気象業務支援センター. メソ数値予報モデル gpv（msm）, 2022. http://www.jmbsc.or.jp/jp/online/file/f-online10200.html, (参照 2024-01-16).

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

[25] Google Research. Colaboratory, 2023. https://research.google.com/colaboratory/faq.html, (参照 2022-03-24).

[26] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, 2015.

[27] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 448–456. JMLR.org, 2015.