画像認識技術による
火星大気シミュレーションデータの解析

師 智薫

神戸大学 理学部 惑星学科
流体地球物理学教育研究分野

2025/03/25

要旨

本論文では, 機械学習による解析の試みとして畳み込みニューラルネットワーク (Convolutional Neural Network, CNN) による画像認識技術を用い, 火星大気のシミュレーションデータの解析を行う. 火星 SCALE-GM を用いて計算した鉛直風のシミュレーションデータから鉛直対流を認識するための CNN を作成し, モデルの学習とモデルによる識別を行った. 学習したモデルを使用し, 鉛直風のシミュレーションデータから画像認識手法による鉛直対流が存在するかどうかの識別を行うことができた.

目 次

第1章　はじめに

この章では, 本論文の研究背景と目的を明示し, 論文で取り扱う鉛直対流について述べる.

1.1　研究の背景と目的

大型計算機の性能の向上により, 解像度の高いシミュレーションデータが得られるようになった. 人の手による大量のデータの解析には量的限界が存在し, 解析のために膨大な時間をかけることもあり得るだろう. このような課題を解決するために, 機械学習の技術を用い人間と同等の認識能力を有した学習モデルによるより効率的な解析を実現することを目指す. 本論文では, 機械学習による解析の試みとして火星の鉛直風のシミュレーションデータを用いた鉛直対流が存在するかどうかを識別するための画像認識を行っていく.

1.2　火星の鉛直対流

ある高さにある空気塊を 1000 hPa まで断熱的に移動させた時の絶対温度を温位という. この温位の勾配が大気の鉛直方向の時に負の値であるときに、熱の輸送を行うために下層の温位が高い空気が上昇し上層の温位が低い空気が下降する図 1.1 のような鉛直対流が発生する. 本研究では, この熱的な要因によって発生する鉛直対流を識別の対象として扱う.

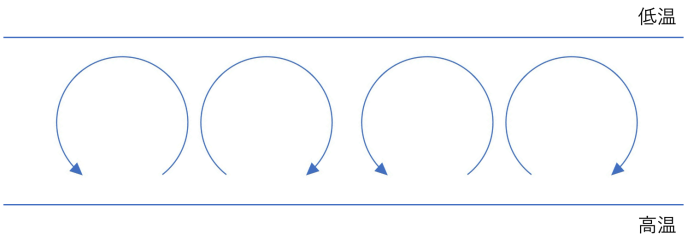

図 1.1: 鉛直対流の模式図. 下層の温位が高い空気が上昇し, 上層の温位が低い空気が下降する.

1.3 CNN による鉛直対流の推測

上で述べたように, 鉛直対流は大気の温位勾配のデータがあれば存在を確かめることが可能である. しかし, 温位勾配のデータが存在しない場合, 鉛直風や密度, 圧力のデータから鉛直対流が発生しているかどうかを推測することができる. 本論文では, 図 1.2 に表されるような全球の鉛直風のシミュレーションデータを CNN による画像認識手法で解析することで, 温位勾配のデータが存在しない場合でも鉛直対流が領域内で存在しているかどうかを推測する手法を述べる. 次章では, CNN に関する基礎知識として機械学習とその仕組みについて説明する.

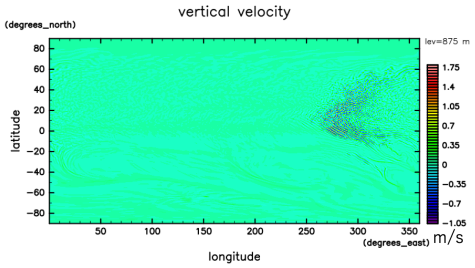

図 1.2: 温位勾配が負の時に発生する鉛直対流による全球の鉛直風. 縦軸は緯度, 横軸は経度, 鉛直風の単位は m/s. 鉛直対流は水平規模が数 km の, スケールの小さい大気現象である. この図の鉛直風は解像度が glevel8 (水平格子間隔 26 km) の時に計算できる鉛直風である.

1.4 本論文の構成

本論文の構成は以下の通りになっている. 第 2 章では機械学習について, 第 3 章では, 解析に用いたデータを述べる. 第 4 章では解析に用いた手法, 構築した CNN について説明する. 第 5 章では, 学習した CNN による画像の推測の結果を, 第 6 章では考察を述べる.

第2章　機械学習とニューラルネットワーク

2.1　人工知能

機械学習を含む人工知能 (Artificial Intelligence) の定義は研究者ごとに異なっている．これは『知性』や『知能』自体の定義がそもそも定まっていないためであり, したがって人工知能の統一した定義を行うことも困難であると考えられているからである．[*1] 本論文では, 人工知能を「人間と同じような知的処理を行うことのできる技術や機械」と定義している．学習を行ったコンピュータが人間と同じように画像の中にあるものを識別ができるようになることを目指す．また, 人工知能は発展段階に着目した分類がされている．以下で紹介する機械学習と深層学習は人工知能の発展段階に応じた分類である．

2.2　機械学習

機械学習 (Machine Learning, ML) とは, 人工知能の分類の 1 つである．コンピュータが人間に近い高度な認識能力を得るためには, パラメータと呼ばれる認識のための基準が必要となる．人間と同じようにコンピュータがデータを元に学習することでパラメータを自動的に決定, すなわち学習するための理論体系が機械学習である．

[*1] 総務省. “平成 28 年度版情報通信白書”. 2016, https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h28/pdf/n4200000.pdf より.

2.3　深層学習 (ディープラーニング)

深層学習 (ディープラーニング) とは人工知能の分類の 1 つであり, 機械学習の発展的な手法である. 従来の機械学習では, 学習を行う際, コンピュータがデータを識別するのに必要な特徴量と呼ばれる数値を人間が設定する必要があった. これは, モデルにとって最適な特徴量であるとは限らないため, 学習精度を向上させる上で課題となっていた. しかし, 深層学習では特徴量をコンピュータ自身が決定する. したがって, よりコンピュータがデータの特徴を捉えやすい特徴量を設定し, 高い認識精度を実現することができる. 深層学習では, モデルが自ら特徴量を設定することでより効率的な学習を行うことができる. 特に画像認識の分野では人間が行うより正確に, かつ膨大な枚数の画像を扱うことができる.

2.3.1　ニューラルネットワーク

ニューラルネットワークとは人間の脳の神経回路を模した形状のモデルであり, 深層学習の基本的な仕組みとなっている. 図 2.1 で示される人間の神経回路は, ニューロンと呼ばれる神経細胞の集まりであり, ニューロン間の信号の伝達によって情報を処理する. このニューロンを模した単一のモデルをパーセプトロンと呼ぶ. 図 2.2 で丸で示したところをノードといい, パーセプトロンのこのノードからノードへの伝達の際に, ノード間の結び付きの強さ (重み) をかけ, その総和と 1 に重みをかけたバイアス項 (定数項) の和に活性化関数をかけることで次のノードへと入力される. パーセプトロンをつなげていくことで, 入力層と出力層の間にいくつも枝分かれをした隠れ層を持つネットワークが完成する. このネットワークをニューラルネットワークと呼ぶ. ニューラルネットワークの隠れ層を増やしていくことでより複雑な出力が可能なディープニューラルネットワークとなる.

[*2]チーム・カルボ. 物体・画像認識と時系列データ処理入門. 株式会社秀和システム, 2021, 584p. 978-4-798-06354-6. より引用.

[*3]筒井　秀和, 岡村　康司. “ランヴィエ絞輪”. 脳科学辞典. 2020-01-09, https://bsd.neuroinf.jp/wiki/%E3%83%A9%E3%83%B3%E3%83%B4%E3%82%A3%E3%82%A8%E7%B5%9E%E8%BC%AA を参考にした.

[*4]吉村　武, 池中　一裕. “髄鞘”. 脳科学辞典. 2014-06-26, https://bsd.neuroinf.jp/wiki/%E9%AB%84%E9%9E%98 を参考にした.

[*5]寺田　純雄, 川岸　将彦. “軸索”. 脳科学辞典.2022-01-05, https://bsd.neuroinf.jp/wiki/%E8%BB%B8%E7%B4%A2 を参考にした.

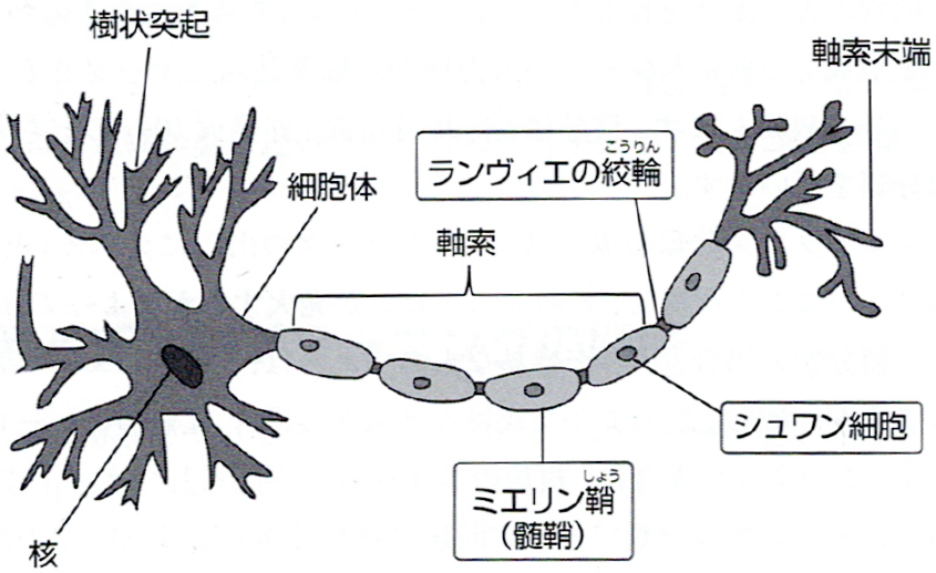

図 2.1:　人間の神経細胞 (ニューロン) の模式図[*2]. 細胞の中心には核があり, 細胞体から伸びる軸索は神経細胞が受けた刺激を他の細胞へと出力を担う器官である. 軸索を取り巻くミエリン鞘 (髄鞘) はシュワン細胞と呼ばれる細胞で構成され, ミエリン鞘があることで信号の伝達速度が上昇する. 軸索がミエリン鞘で囲まれていない間隙をランヴィエの絞輪と呼ぶ. ニューロンは別のニューロンへと信号を伝え, 樹状突起で他の細胞からの信号入力を受け取る. ニューロンが繋がったネットワークを神経回路という[*3*4*5].

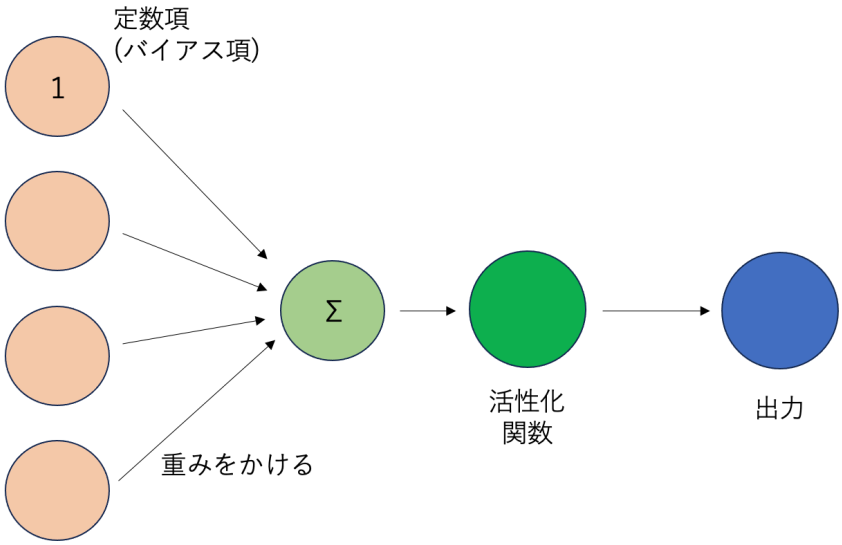

図 2.2:　パーセプトロンの模式図. 各ノード (図中の丸で示した部分) の入力に重みをかけ, その総和と 1 に重みをかけたバイアス項 (定数項) の和に活性化関数をかけることで次のノードへと入力される仕組みとなっている.

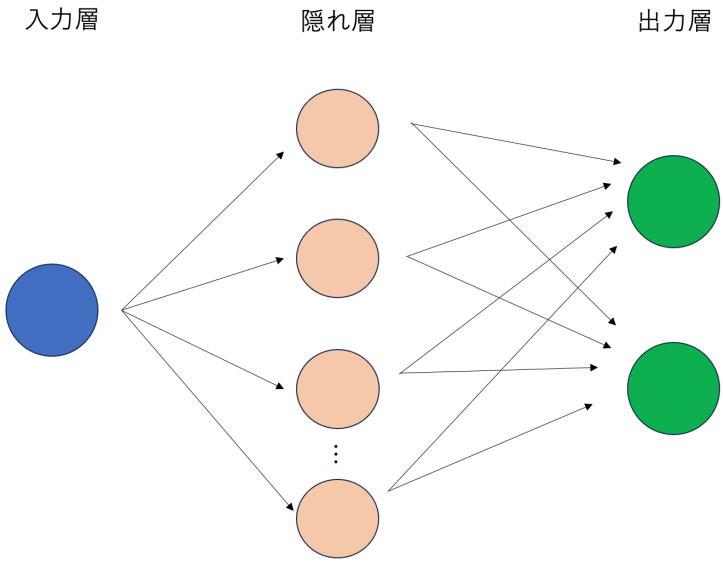

図 2.3:　ニューラルネットワークの模式図. パーセプトロンを繋げていくことで図のような形状のモデルとなる.

2.3.2 損失関数

学習モデルによる出力と学習モデルが目指す正解が, どの程度離れているかの指標となる関数を損失関数 (損失) という. 本論文では, 交差エントロピー誤差 (クロスエントロピー誤差) を損失関数として使用する. 交差エントロピー誤差 Q は, 学習モデルによる出力が $\hat{z}_c$, 正解の出力が z_cの時, $Q = -\sum_{c=1}^{C}\sum_{k=1}^{K} z_c(k)\log\hat{z}_c(k)$ (2.1) で表される. 損失関数モデルが正解に近づくためには, この損失関数の値が小さくなるように重みを更新し最適解を求める必要がある. 学習モデルによる出力と正解となる出力の差は学習時とは逆方向, すなわち出力層から入力層に向かって伝播され, 伝播された差によって各ノードは重みを更新する. 本論文で作成する学習モデルは確率的勾配降下法 (Stochastic Gradient Descent, SGD) と呼ばれるアルゴリズムを用いて重みを最適解に近づけている.

確率的勾配降下法

図 2.4 のように損失関数を縦軸, 重みとバイアスを横軸とした時, 損失関数が 0 に近づくほど勾配は小さくなる. このことを利用し, 勾配を求め重みを更新していくことで, 最適解に近づけていく方法を勾配降下法という. 勾配降下法のうち, 確率的勾配降下法でははランダムに取り出した 1 つのデータを用いて勾配を求め, パラメータを更新していく作業をデータの数だけ行う. 現在の学習が k 回目の時, 損失関数を Q とすると, 入力のノードが M 個の時, 各ノードの入力にかけられる重み $w_m(m = 1, 2, ..., M)$ の最適解は以下の手順で見つけることができる.

初期値：

$$w_m(0) = (\text{適切な任意値})(m = 1, 2, ..., M) \tag{2.2}$$

繰り返し処理：for $k = 0, 1, ..., K$:

$$\Delta(k) = \frac{\partial Q}{\partial w_m}(w_1(k), w_2(k), ..., w_M(k)) \tag{2.3}$$

$$w_m(k+1) = w_m(k) - \eta\Delta(k) \tag{2.4}$$

式中の η は学習率であり, η の値を変更することで一度にどの程度更新するかを調整できる. また, $\eta > 0$ である.

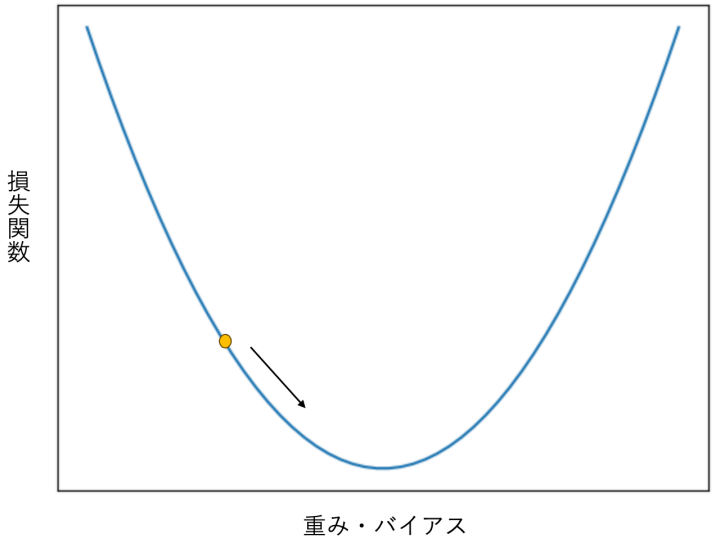

図 2.4:　勾配降下法の模式図. 損失関数を縦軸, 重みとバイアスを横軸とした時, 損失関数が 0 に近づくほど勾配は小さくなる.

2.4　本論文の構成

本論文の構成は以下の通りになっている. 第 2 章では, 解析に用いたデータを述べる. 第 3 章では解析に用いた手法, 構築した CNN について説明する. 第 4 章では, 学習した CNN による画像の推測の結果を, 第 5 章では考察を述べる.

第3章　使用データ

3.1　使用データ

本論文で使用したデータは火星の大気大循環モデルである火星 SCALE-GM で火星大気のシミュレーションを行った鉛直風のデータである.

3.1.1　火星 SCALE-GM

火星 SCALE-GM は地球の大気大循環モデル SCALE-GM[*1] より. を元に作られた火星の大気大循環モデルである. 計算機の性能が向上し高解像度の計算が可能となったことから, 水平数 10 km 規模以下のスケールで運動を表現できるようになった. 水平 10 km 以下のスケールでは従来の全球モデルで用いられる静力学的な方程式が成り立たない, そのため, 火星　SCALE-GM では計算に非静力学系の方程式を用いており, すなわち水平規模が数キロメートルの現象も表現できる高解像度全球大気計算が可能となっている.

3.1.2　計算条件

シミュレーション時の水平解像度は glevel8, すなわち水平格子間隔が 26km, 地形無し条件の元で北半球春分から約 60 日計算したデータとなっている. 緯度, 経度, 高度, 時間の次元を持ち, 単位が m/s の鉛直風の全球データを使用した.

[*1] 理化学研究所計算科学研究センター, SCALE Global Model (SCALE-GM). https://scale.riken.jp/ が開発.

3.2 データの加工

シミュレーションデータは緯度, 経度, 高度, 時間の次元を持つ鉛直風の多次元配列のデータとなっているため, 画像認識を行うために 2 次元の画像データへと加工をする必要がある. また, モデルの学習では学習に用いるデータ (学習データ) と学習した後に予測ができるかどうかを確かめるためのデータ (テストデータ) を用意する. 学習データとテストデータに事前に正解 (ラベル) をつけておくことで, モデルは画像と正解を照らし合わせながら学習を行うことができる. また, 学習を終えたモデルにテストデータを使用して予測をさせることで, モデルの性能を評価することができる. 本論文ではプログラミング言語に Python を用いて加工を行った.

3.2.1 データの分割

はじめに, 全球のデータを識別のための最小単位として 7.5°×7.5° の領域に分割した. その後, 鉛直風の多次元配列のデータを 30 × 30 ピクセルの 2 次元の画像データへと変換した. この時, 画像の視認性を上げるためにカラーバーの範囲を 0 m/s から 1 m/s に変更している. 本論文では, 7.5°×7.5° の領域に分割されたそれぞれの画像の中に鉛直対流があるか, あるいはないかを判断するための 2 つのグループに分類するモデル (二値分類モデル) を作成した. したがって, 図 2.3 のように鉛直対流がある領域のデータを「1」, 図 2.4 のように鉛直対流がない領域のデータを「0」とラベルを分け, 画像 1 枚ずつに次節で説明する方法でラベル付けを行った. これにより, 執筆者が行った人間の目による分類をモデルに学習させることができる.

3.2.2 データのラベル付け

ラベル付けは, 各ラベルごとにフォルダにまとめた後に, 各画像ごとの 0 もしくは 1 のラベルで分類したフォルダ名を抽出し, 新たなデータセットとして作成する. これにより, 分割された鉛直風の画像データとそれに対応したラベルの 2 つのデータセットを用意することができる. 学習データは, 高度 1km で切り出したシミュレーションの計算最終時刻までの 10 火星日間, 全球を 7.5°×7.5° の領域に分割し, 1 枚ずつラベル付けを行った計 92160 枚の 30 × 30 ピクセルの データを用意した. テストデータは計算最終時刻の瞬間場を高度 1125 m で切り出したものを, 学習データと同様に 7.5°×7.5° の領域に分割した後ラベル付けをした 1152 枚

のデータを用いた. 作成した画像は学習モデルに入力するために 1 ピクセルごとに RGB の 3 値で表される画像から明度を 1 値で表すグレースケールの画像に変換している.

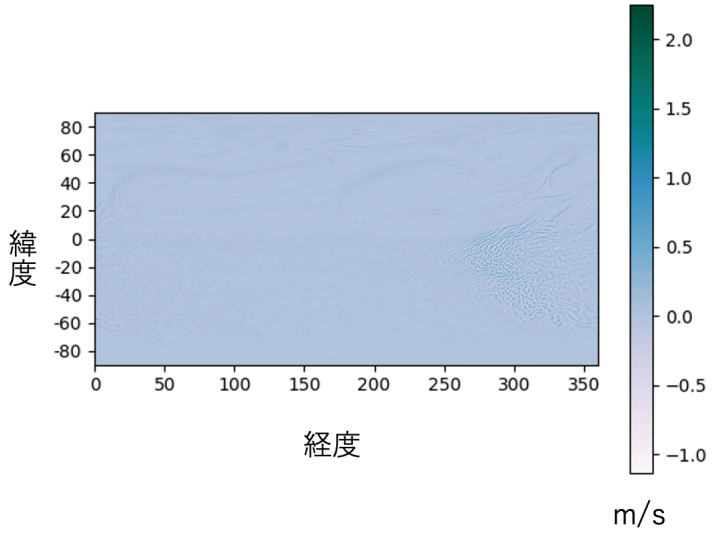

図 3.1: 全球の鉛直風を描画した画像. 縦軸は緯度, 横軸は経度を示す.

3.2.3 オーバーサンプリング

全球を描画した画像である図 2.1, 図 2.2 を見ると, 鉛直風がある範囲は全球に比べて面積が大幅に狭く, ラベル付けを行った時に「1」のラベルがついたデータが「0」のデータに比べ少なくなってしまうことがわかる. このようなデータの不均衡は学習の精度の低下を招く. したがって, 学習の精度を上げるために少ないラベルのデータを増やす「オーバーサンプリング」によるデータの均衡化を行った. 学習データの中から鉛直対流がある「1」のラベルがついたデータに回転や左右反転, 上下反転の処理を加え, 92160 枚から 148160 枚に全体の学習データを増やすことで, 不均衡による学習精度の低下が起こらないようにした.

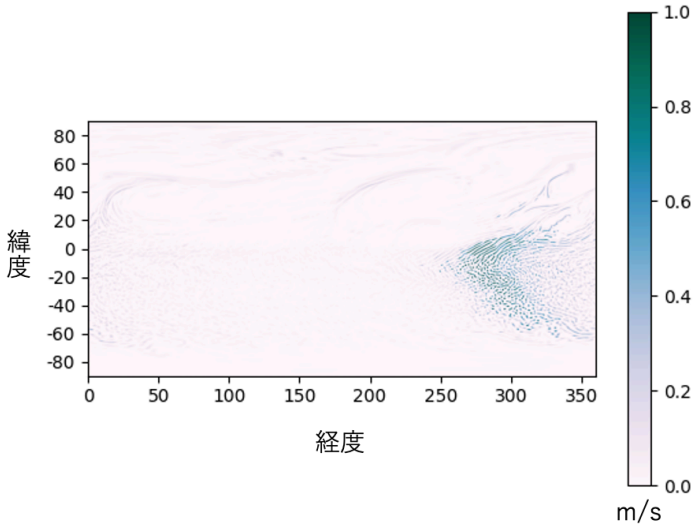

図 3.2: 全球の鉛直風を描画した画像. 視認性を上げるためにカラーバーの範囲を 0 m/s から 1 m/s に変更している. 鉛直対流と見られる強い鉛直風が画像右端と左端で発生している.

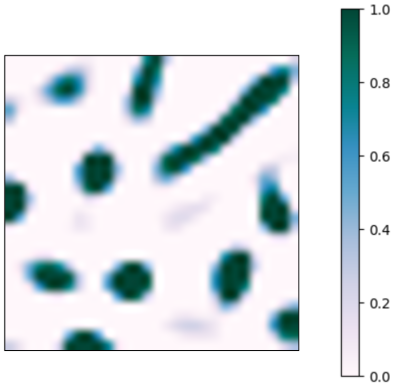

図 3.3: 30×30 ピクセルで分割した画像の一例. 鉛直風が確認できるため, 「1」のラベルをつける.

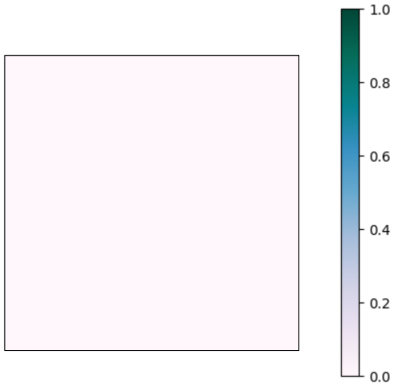

図 3.4: 30×30 ピクセルで分割した画像の一例. 強い鉛直風は確認できない. 「0」のラベルをつける.

第4章　解析手法

4.1　畳み込みニューラルネットワーク (CNN)

畳み込みニューラルネットワーク (Convolutional Neural Network, CNN) とは, 多次元配列に特化したニューラルネットワークである. 通常のニューラルネットワークでは 2 次元の画像データなども 1 次元の配列に変換して学習を行うが, 図 4.1 で示すように CNN では 1 つのニューロンが 2 次元空間の情報, すなわち画像のピクセル同士の位置関係の情報を保持したまま学習を行うことができるため, 画像認識の分野において活用される.

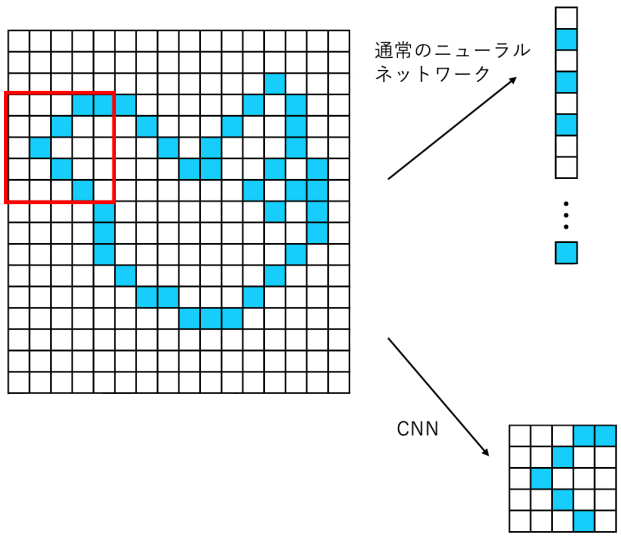

図 4.1:　CNN の学習の模式図. 通常のニューラルネットワークでは赤線で囲った部分を 1 次元の配列に変換するが, CNN では囲われた部分の位置関係を保ったまま学習を行う.

4.1.1 CNN の構築

CNN は入力された画像に対し特定の演算を行うための処理方法を持ち, これをフィルタという. また, CNN も通常のニューラルネットワークと同じように複数の層を持ち, 本論文で使用する CNN は入力層, 畳み込み層, Flatten 層, 出力層で構成されている. 畳み込み層ではフィルタを複数用いて, 画像の中の特徴的な部分を強調する処理を行う. Flatten 層では畳み込み層から出力した多次元配列を 1 次元配列に変換する. 出力層は分類するグループの数によってノードの数が変化し, 各グループの確率を出力する. 本研究では, Python の ディープラーニング用のライブラリである tensorflow.keras を使用し, 鉛直対流が存在する場所と存在しない場所の 2 クラスに分類するための CNN モデルを作成した.

学習の流れ

本論文で扱う CNN はミニバッチ勾配降下法と呼ばれる手法で畳み込み層の各フィルタでの入力や出力層の入力にかけられる重みの更新を行う. これは, 学習データから無作為に抽出した少量のデータを用いて確率的勾配降下法による重みの更新を行い, 全てのデータを使い切るまで少量のデータによる学習を繰り返す手法である. ミニバッチ 1 つあたりのデータ数) ×ミニバッチの数 = 学習データ全体の数となり, 学習 1 回 (1 エポック) につきミニバッチの数だけ重みの更新が行われる.

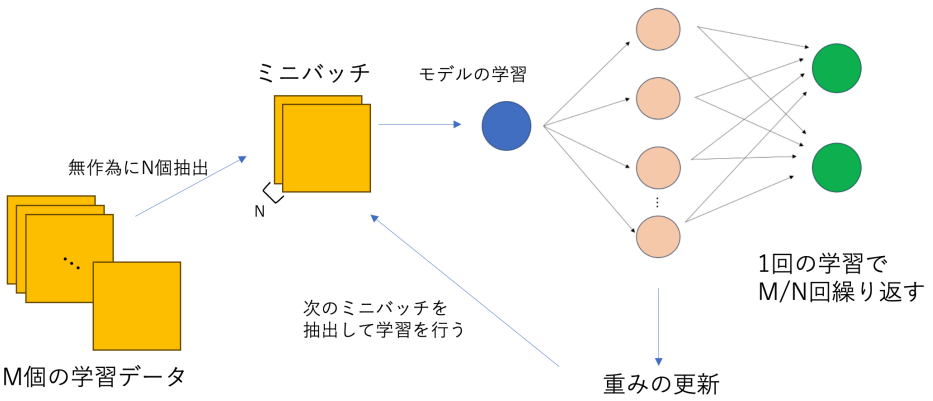

図 4.2: ミニバッチ勾配降下法の模式図. 学習データ全体を M, ミニバッチのサイズを N とすると, 1 エポックにつき M/N 回の重みの更新が行われる.

入力層

入力層では, 画像データを (データサイズ, 行データ, 列データ, チャネル) の 4 次元配列に変換して入力を行う. データサイズは画像データの枚数, 行データと列データは 1 枚の画像の縦横のサイズを表す. チャネルは 1 ピクセルあたりの RGB などのピクセル値を格納するための次元であり, 本論文では 1 値のグレースケールの明るさの値が格納される. したがって, 本論文で作成するモデルでは入力される配列が (148160, 30, 30, 1) となる.

畳み込み層

画像内の特定の形状に反応するフィルタを入力されたデータに重ね, 検出する処理を行うための層となる. フィルタのサイズは 3×3 ピクセルの 2 次元フィルタである. 図 4.3 に一例を示すようにこのフィルタを 30×30 の画像のある部分に重ね, 画像の値とフィルタの積の和を求め, 元の画像の中心に書き込んでいく. この作業をフィルタを画像全体にスライドさせながら繰り返すことで, フィルタが反応した画像内の特定の形状が画像内のどの位置にあるかを出力することができる. この畳み込み演算 (Convolution) と呼ばれる処理を行うことで, 画像内の特徴的な部分を検出し強調する. 畳み込み演算を一般化した式は, 画像の位置 (i, j) のピクセルの値を $x(i, j)$ フィルタの値を $h(i, j)$, 畳み込み演算で得られる値を $c(i, j)$ とした時,

$$c(i, j) = \sum_{i,j}^{n} x(i, j) \cdot h(i, j) \tag{4.1}$$

で表される. ニューラルネットワークにおける重みはこのフィルタの各ピクセルにかけられる. 図 4.4 のようにフィルタの重みを更新することで分類に最適な形状を抽出できるようになる. 本論文のモデルで使用する使用するフィルタは 32 枚, このフィルタの数が畳み込み層のノード数となる. フィルタの数とフィルタのサイズをかけた数値が畳み込み層の重みの数となる. すなわち, 本論文のモデルの畳み込み層は $3 \times 3 \times 32 = 288$ 個の重みを持ち, さらに各フィルタごとに 1 つずつ定数項が存在する. フィルタ (ノード) と通した時の活性化関数は, 以下の式で表される ReLU(Rectified Linear Unit) 関数 (正規化線形関数) を用いる. これは入力が 0 を超えていれば入力された値をそのまま出力し, 入力が 0 以下であれば 0 を出力する関数である. CNN を含むニューラルネットワークでは, 重みの更新に使う勾配が層を通過する度に掛け合わされていく. 複数の層で勾配の値が小さい時, 掛け合わされることで勾配が 0 に近づく勾配消失問題が発生する. ReLU 関数は

[*7] チーム・カルボ. 物体・画像認識と時系列データ処理入門. 株式会社秀和システム, 2021, 584p. 978-4-798-06354-6. を参考に作成.

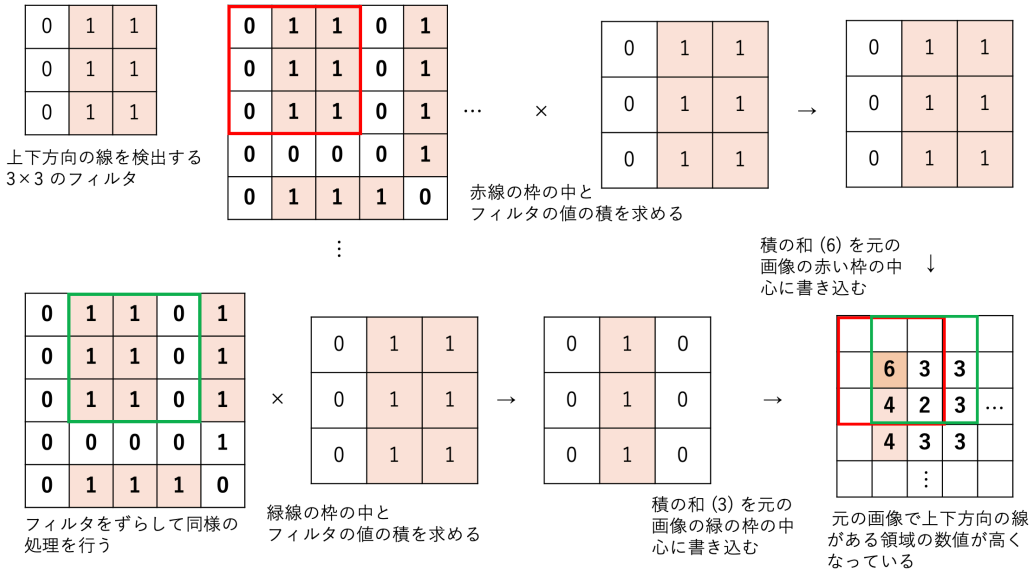

図 4.3: フィルタを用いた畳み込み演算の一例の模式図 [*7]. 上下方向の線を検出する 3 × 3 の大きさのフィルタと入力された画像の赤線で囲まれた部分を重ね, 各ピクセルごとの積を求める. そして積の和である 6 を赤線で囲まれた領域の中心に書き込む. この一連の処理をフィルタを少しずつずらして繰り返すことで, 元の画像の中で上下方向の線がある部分が強く検出される.

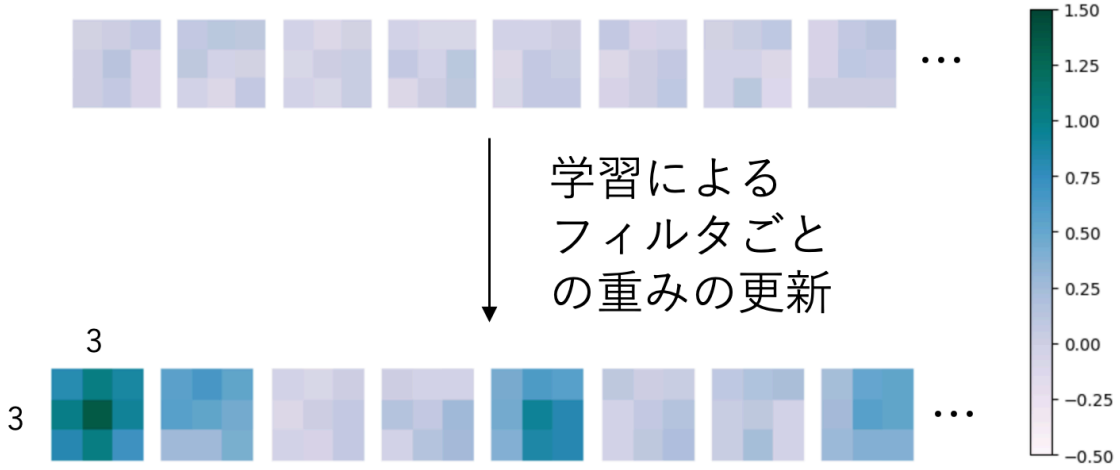

図 4.4: フィルタの 1 ピクセルあたりの重みを可視化した模式図. 学習を行うことで各ピクセルの重みが更新される.

入力が 0 以上ならば勾配が 1 となるため, 活性化関数として ReLU 関数を使用することで損失関数の勾配消失を防ぎ, 学習を進みやすくすることができる.

$$ReLU(x) = \begin{cases} x(x > 0) \\ 0(x \leq 0) \end{cases} \tag{4.2}$$

また, 入力データの幅を w, 高さを h, フィルタの幅を fw, 高さを fh とおいた時, 出力されるデータの幅は $w - fw + 1$, 出力される高さは $h - fh + 1$ となり, 畳み込み演算を行うことで入力されたデータのサイズより出力されるデータのサイズが小さくなる. 図 4.3 でも畳み込み演算を行った後出力される画像は元の画像より幅と高さがそれぞれ 1 ピクセル分小さくなっている. 本論文で作成したモデルでは畳み込み演算によるサイズの減少を防ぐためにゼロパディングと呼ばれる手法を用いている. これは, 元の画像の周りをゼロで囲むことで画像を一回り大きくしてからフィルタによる演算を行う手法である. ゼロパディングにより, 出力される画像は元の画像と同じサイズになり画像の端の情報が出力に反映されやすくなる. ゼロパディングを行っているため, 元の画像の大きさが 30×30 ピクセルの時, 32 枚のフィルタを通した出力は幅と高さが変わらない画像 32 枚の 2 次元の配列データとなる. したがって出力の配列は (30, 30, 32) の 3 次元配列となる.

Flatten 層

畳み込み層からの出力である (30, 30, 32) の 3 次元配列を 1 次元配列に変換するための層が Flatten 層である. これは最終的な出力が 1 次元で行われるためである. Flatten 層を通すことで (30, 30, 32) の 3 次元配列は $30 \times 30 \times 32 = 28800$ の 1 次元配列となる.

出力層

本論文では二値分類を行うため, 出力層のニューロンの数は 2, 重みの数は $28800 \times 2 = 57600$ となる. 出力層でも活性化関数が適用され, 以下の式で表されるソフトマックス関数を使用する. 分類したい各グループの確率を 0 から 1.0 の間の実数で出力し, 出力値の総和は 1 となる関数であり, ノード数が n 個の時, k 番目の出力 y_k は

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n} \exp(a_i)} \tag{4.3}$$

となる. ソフトマックス関数を用いることで, 1 つ目のグループが正解である確率は 30 %, 2 つ目のグループが正解である確率は 70 % という形で出力を確立的に解釈することができる.

作成した CNN モデルの概要を表 4.1 に, モデルの模式図を図 4.5 に示す.

表 4.1: 作成したモデルの概要

層の種類	出力されるデータの形状	定数項を含むパラメータ数
畳み込み層	(30, 30, 32) の 3 次元配列	320
Flatten 層	28800 (1 次元配列)	0
出力層	2 (1 次元配列)	57602

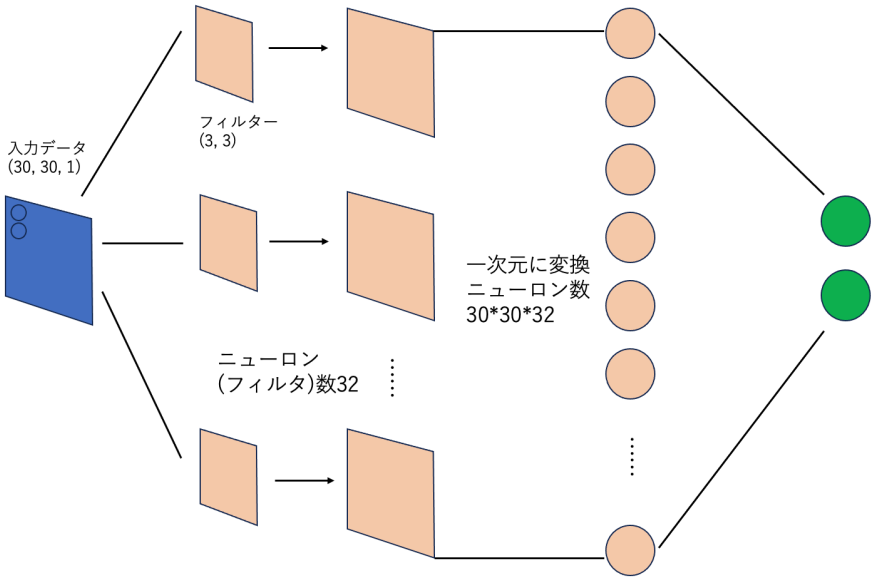

図 4.5:　作成した CNN モデルの模式図. 左から順に入力層, 畳み込み層, Flatten 層, 出力層となっている. [*8]

4.1.2　検証データ

148160 枚の学習データのうち 20% にあたる 29632 枚の画像は学習に使わず, モデルが未知のデータに対応できるかどうかを確かめるための検証データとして使う. この検証データ用の画像は 1 回の学習ごとにランダムに選ばれ, その回の学習

[*8] チーム・カルボ. 物体・画像認識と時系列データ処理入門. 株式会社秀和システム, 2021, 584p. 978-4-798-06354-6. を参考に作成.

には使われない. 残りの 118528 枚を 1 回の学習で学習データとして使用し, この学習データから少量のデータを 1 つのミニバッチとして無作為に抽出しながら学習を行う. 今回は 128 枚のデータを 1 つのミニバッチとして扱い, 合計 926 個のミニバッチを 1 回の学習で用いる.

4.1.3 損失と正解率

前章で述べたように, 正解の出力と学習モデルの出力である損失を減らすように重みを更新することでモデルは正解の出力に近づく. 正解率は (予想と正解が一致するデータの数)/(全体のデータの数) で求めることができる. 正解率も損失と同様に学習の精度を評価する基準となる.

4.1.4 早期終了

本論文で作成するモデルは学習回数の上限を 100 回に設定した. しかし, 学習回数が 100 に達する前に, モデルの精度が限界まで向上しそれ以上改善しない状態になることがある. そういった場合に, 学習を 100 よりも手前で終了するように設定する. 本論文のモデルでは損失の値を早期終了のための基準とし, 損失の値が 5 回以上改善されなかった場合に学習を終了する.

4.1.5 ドロップアウト

ディープラーニングでは学習を繰り返し行うことで学習データに過剰に適合し, その結果未知のデータへの認識の精度が低下する過学習 (オーバーフィッティング, 過剰適合) が発生することがある. 過学習を防ぐために, 本論文で作成するモデルではドロップアウトと呼ばれる処理を行っている. これは, 特定の層に含まれるノードを一定の確率でランダムに選び, 選ばれたノードを無効にするという手法である. ドロップアウトを行うことで, ミニバッチごとに違うノードがランダムに無効化され過学習を防ぐことができる. さらに, ノードのランダムな無効化により複数のニューラルネットワークを別々に学習させ, 組み合わせて予測を行うのと同様の効果を得ることができる.

4.2 テストデータを使った識別

本論文では学習が終了した後, 学習済みのモデルを使用して未知のデータへの識別を行う. 未知のデータとして第 3 章で用意したテストデータを使用する. テストデータは図 4.6 で示す. テストデータにも学習データと同様に 7.5°×7.5° に分割した領域内に鉛直対流による鉛直風が存在するか, もしくはしないかの 2 つのグループに分類し, それぞれラベルをつけてモデルの予測との比較を行う. 分割してラベル付けを行った各領域を元の全球のデータに並べ替えたものが図 4.7 である. テストデータを使った識別の結果は次章で示す.

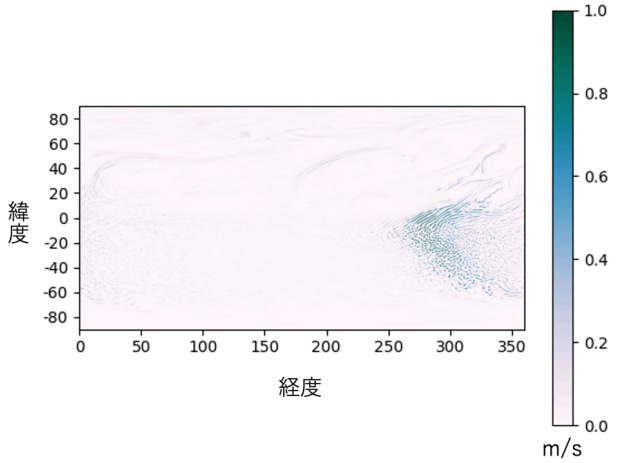

図 4.6: テストデータとして用意した, 計算最終時刻の瞬間場を高度 1125 m で切り出した全球の鉛直風をカラーバーの範囲を 0-1 m/s に変更して描画したもの. 横軸は経度, 縦軸は緯度を示す. 画像右端と左端に鉛直対流と見られる鉛直風が確認できる.

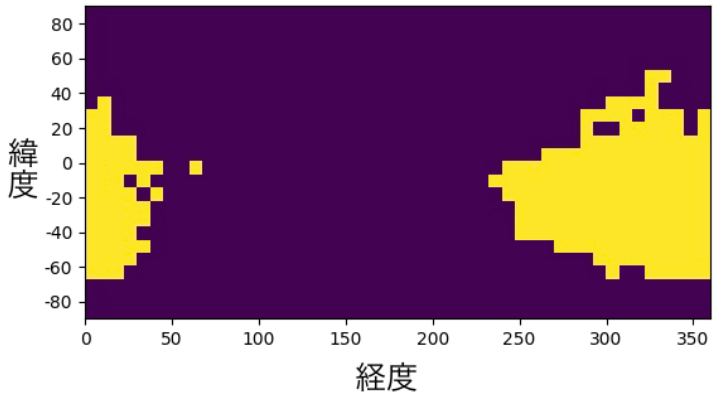

図 4.7: テストデータの正解のラベルを描画した画像. 横軸は経度, 縦軸は緯度を示す. 領域内に鉛直対流による鉛直風がない部分である「0」のラベルを紫, 鉛直風がある部分である「1」のラベルを黄色で描画している. この図は分割した各領域を元の全球の状態に並べ替えたものである.

第5章　結果

5.1　学習の推移

学習は 18 回目で早期終了した. 検証データによる最終的な正解率は 81.46%, テストデータによる最終的な正解率は, 95.31% である. 学習の各回ごとにどのように正解率と損失が変化したかを図 4.1 に示す. このグラフは学習の推移の時系列データを示したものであり, 左のグラフが損失, 右のグラフが正解率である. 学習が進むにつれて損失が減少し, 正解率が上昇していることがグラフから読み取れる.

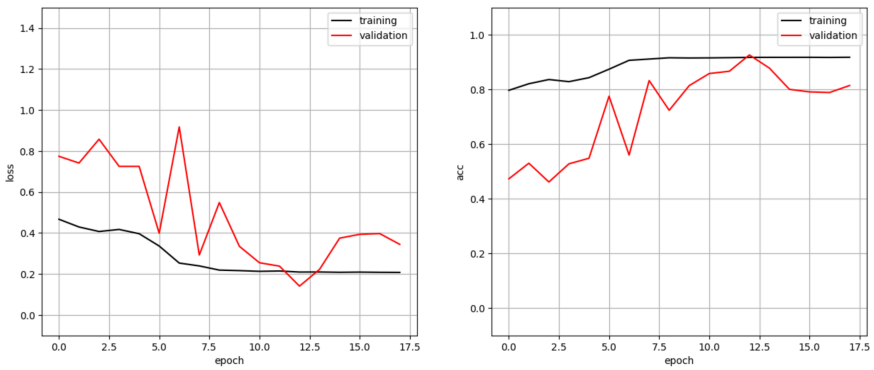

図 5.1:　学習の推移の時系列データを示したグラフ. 左が損失, 右が正解率である. 赤線は検証データ, 黒線は訓練データの推移を示す.

5.2　学習したモデルによる識別

学習を行ったモデルとテストデータを用いて実際に識別を行った. 学習時と同じようにテストデータ用の画像を分割し, 各領域に鉛直対流があるかどうかを, 鉛直対流がない場合は「0」ある場合は「1」で答える. 以下の表ではモデルによる予測のラベルと正解のラベルの比較を行う. 上段のカッコ内は正解が 0 の画像に対し,

0 と予測した割合, 1 と予測した割合を示す. 下段も同様に, 正解が 1 の画像に対し, どちらのラベルで予測したかの割合を示す. 図 5.2 から図 5.4 では予測したラベルとテストデータの比較を行っている. 学習モデルによる識別が行われていることが確認できるが, 本論文で識別したい鉛直対流による鉛直風ではない画面中央付近の鉛直風も鉛直対流がある部分として識別している.

表 5.1: 予測と正解の比較

予測と正解の比較	予測が 0	予測が 1
正解が 0	916(98.39%)	15(1.61%)
正解が 1	39(17.65%)	182(82.35%)

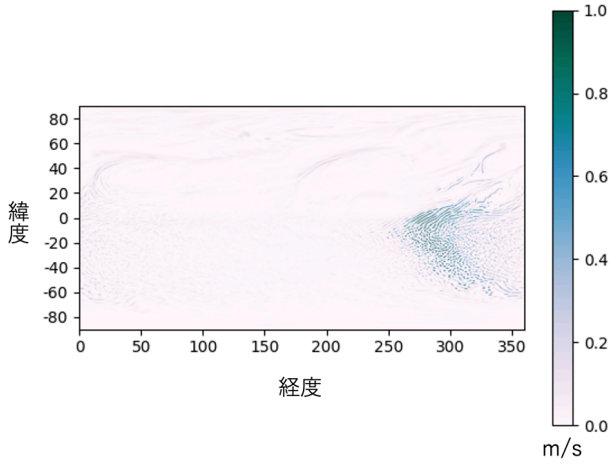

図 5.2: 4 章で説明したテストデータを描画した画像. 横軸は経度, 縦軸は緯度を示す. 画像右端と左端に鉛直対流と見られる鉛直風が確認できる.

図 5.5 を見ると, モデルが鉛直風のシミュレーションデータから鉛直対流が存在するかどうかを識別できていることがわかる. しかし, 緯度 40$\tilde{6}$0 °, 経度 200 ° 付近で本論文で扱う熱的な鉛直対流と関係のない鉛直風を鉛直対流が存在する領域として識別していることがこの図から読み取れる. このように, 鉛直対流とは直接関係のない鉛直風の画像データも鉛直流がある領域のデータとして識別する

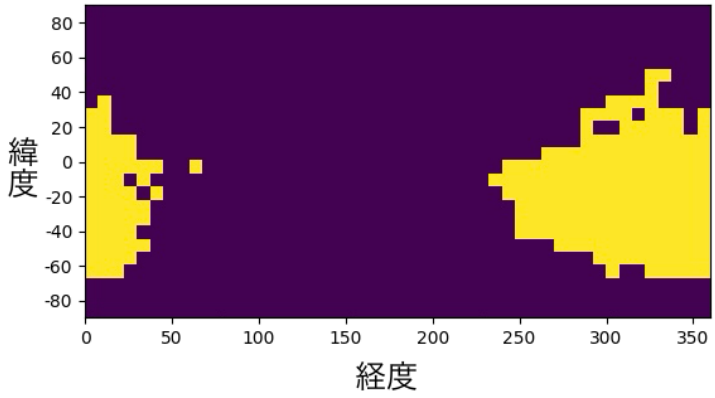

図 5.3: テストデータの正解のラベルを描画した画像. 横軸は経度, 縦軸は緯度を示す. 鉛直対流による鉛直風がない「0」のラベルを紫, 鉛直風がある「1」のラベルを黄色で描画している.

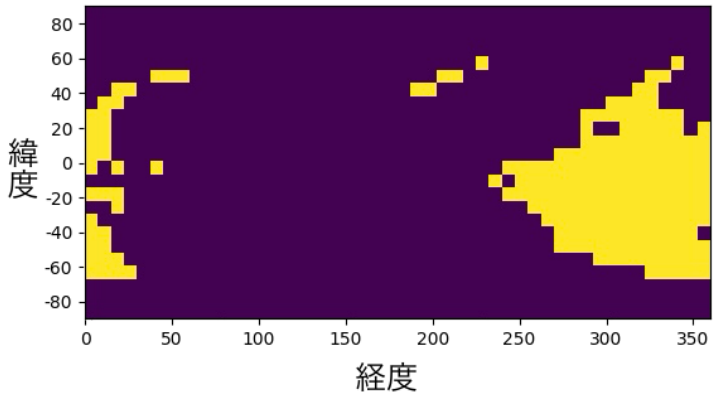

図 5.4: 7.5°×7.5° に分割したテストデータの各領域に鉛直対流があるかどうかをモデルに識別させ, その結果を全球に並べ直し描画した画像. 「0」を紫, 「1」を黄色で描画している. 画面右端と左端を「1」と予測していることがわかる.

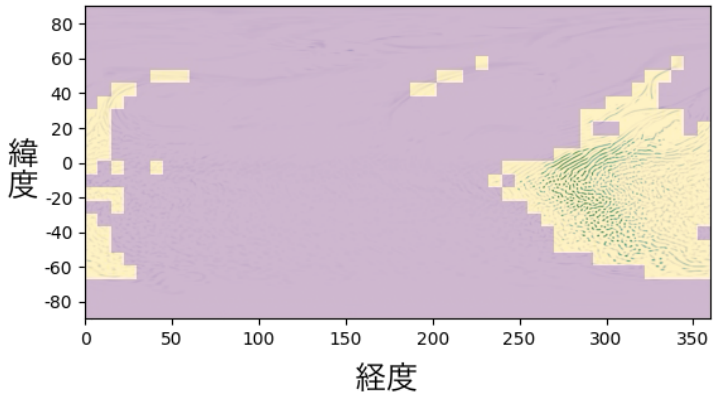

図 5.5: 図 5.2 と図 5.4 を重ねた画像. 鉛直対流による鉛直風が発生している位置とモデルが鉛直対流があると画像から識別した位置が重なっていることがわかる. 緯度 40$\tilde{6}$0 °, 経度 200 ° 付近で鉛直対流による鉛直風が発生している領域を鉛直対流がある領域として認識している.

第6章　考察

6.1　データの不均衡による精度の低下

本論文ではデータの加工の際, オーバーサンプリングにより少ないラベルのデータを増やすことで不均衡を解消し精度を上げて学習を行った. 本節では研究の過程の紹介として, オーバーサンプリングを行わない状態での学習を行いその精度を検証した. データの加工や学習に使ったモデルの条件は変わらず, 学習に使う画像データの枚数をオーバーサンプリングによる均衡化を行った 148160 枚から 不均衡な状態の 92160 枚に削減する. この時, 「0」のラベルがついた画像データは 77643 枚, 「1」のラベルがついた画像データは 14517 枚となる. 学習の後に同様にテストデータを使ったモデルによる予測を行い, 正解との比較を行ったものが表 6.1 である. また, 予測されたラベルを全球に変換したものを図 6.1 に示す. 図の通り, 鉛直対流による鉛直風がある領域のデータが少なかったことで識別の精度が下がり, モデルが鉛直対流が学習できていないことがわかる.

表 6.1: 不均衡データで学習を行った場合の予測と正解の比較

予測と正解の比較	予測が 0	予測が 1
正解が 0	931(100.0%)	0(0.0%)
正解が 1	211(100.0%)	0(0.0%)

6.2　データ数を削減した場合の学習

データ数を減らした場合でも元の同様の精度を維持できれば, 今後の学習でデータ量を削減しより短時間で研究を進めることができると考えられる. 本節では, 元

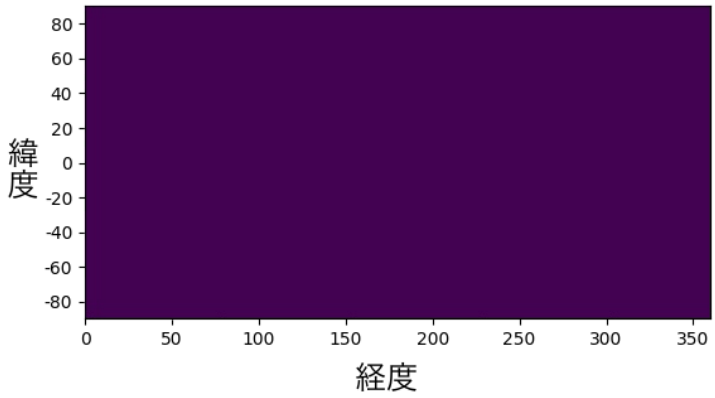

図 6.1: 不均衡なデータによる学習を行ったモデルが予測したラベル. 第 4 章と同じテストデータを使用し黄色が鉛直対流がある「1」のラベル, 紫色が鉛直対流がない「0」のラベルを示す. 縦軸の緯度方向に 24 枚, 横軸の経度方向に 48 枚, 計 1152 の領域全てが「0」のラベルとなっている. これは鉛直対流による鉛直風を識別できていないことを示す.

のデータの 1/2, 1/4 と画像データの枚数を減らした時に識別の精度がどのように変化するかを検証していく.

1/2 に削減した場合の学習

学習データの枚数を元の 148160 枚からその半分である 74080 枚に削減し, 学習を行った. 表 6.2 は予測と正解のラベルの比較を行ったもの, 図 6.2 はモデルによる予測結果を描画した画像である. 元のデータ量 (第 4 章参照) の時と比べ, 左側の鉛直風を識別できなくなっているが, 右側の鉛直対流による強い鉛直風の識別はできていることがわかる.

表 6.2: 半減したデータで学習を行った場合の予測と正解の比較

予測と正解の比較	予測が 0	予測が 1
正解が 0	928(99.68%)	3(0.32%)
正解が 1	80(36.2%)	141(63.80%)

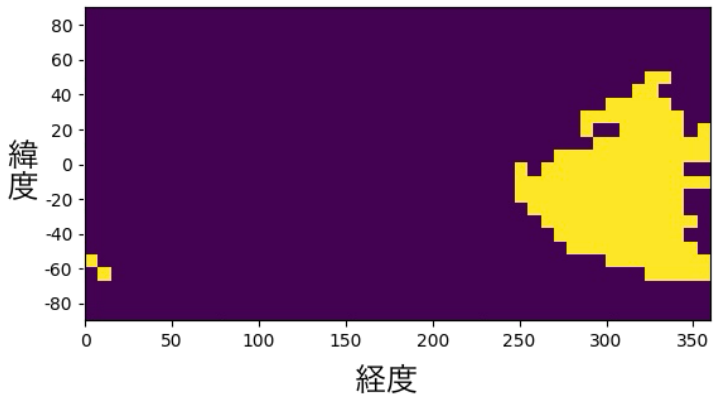

図 6.2: データ量を元のデータの半分に減らして学習を行ったモデルによるラベルの予測結果.

1/4 に削減した場合の学習

学習データの枚数を 74080 枚から更にその半分である 37040 枚に削減し, 学習を行った. 表 6.3 は予測と正解のラベルの比較を行ったもの, 図 6.3 はモデルによる予測結果を描画した画像である. 1/2 に減らした時と大きな差はないことがわかる.

表 6.3: 1/4 に削減したデータで学習を行った場合の予測と正解の比較

予測と正解の比較	予測が 0	予測が 1
正解が 0	929(99.79%)	2(0.21%)
正解が 1	82(37.10%)	139(62.90%)

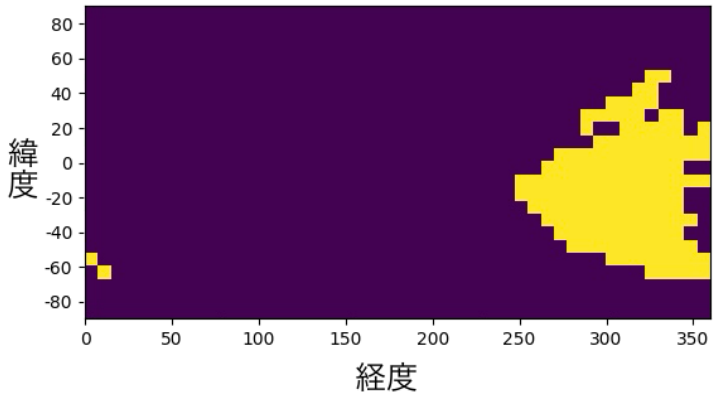

図 6.3: データ量を元のデータの 1/4 に減らして学習を行ったモデルによるラベルの予測結果.

上記の結果から, 各ラベルごとのデータの量が均衡であれば, データの量を 1/4 まで削減して学習を行っても強い鉛直風の識別は可能であると考えられる.

6.3 マルチクラスの分類

前節まで行っていた研究では, 単直な鉛直風の有無による二値分類だったため, 鉛直対流とは直接関係のない鉛直風の画像データも鉛直流がある領域のデータとして識別することが課題となっている. したがって, 鉛直風がある部分のデータをさらに細分化することで, より正確に鉛直対流の識別が実現できると考えられる. 本節では鉛直風のデータを描画した時の形状でさらに分類することで 2 グループから 3 グループへ分類するグループを増やし, 学習を行った際の精度を検証する. 鉛直風のない領域は二値分類の時と同じように「0」, 図 6.4 のように筋状の鉛直風を「1」, そして図 6.5 のように鉛直風を「2」として再度分類を行う. 使用データや加工方法は変えず, モデルの出力層のニューロン数を分類したい数に合わせて 2 から 3 に変更する. また, オーバーサンプリングによるデータの均衡化により, 学習データに使われる画像は合計で 218160 枚となる.

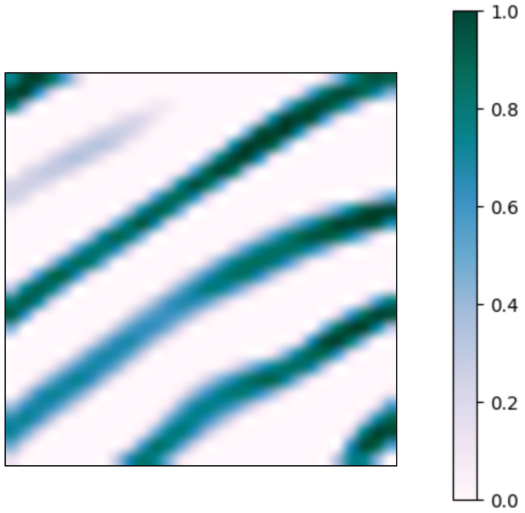

図 6.4: 「1」に分類し直した画像の一例. 筋状に描画された鉛直風が確認できる.

以下に学習を行ったモデルによる識別の結果を示す. テストデータは第 5 章の識別で使用したものと同じデータを用いている. 表はラベルごとの予測と正解を比較し, 各正解のラベルごとの予測結果の割合を出したものである. 図 6.6 は図 4.6 と同様にテストデータの各領域の正解となるラベルを全球に並べ直し描画した画像, 図 6.7 は学習モデルが行った予測のラベルを全球で描画した画像, 図 6.8 はテストデータを描画した画像と学習モデルによる予測のラベルを重ねた画像である. 2 グループでの分類と比較して, 強い鉛直風の有無は識別できているが左端のやや弱い鉛直風の識別ができず, 鉛直風を含む 1 や 2 のラベルの正答率が低下してい

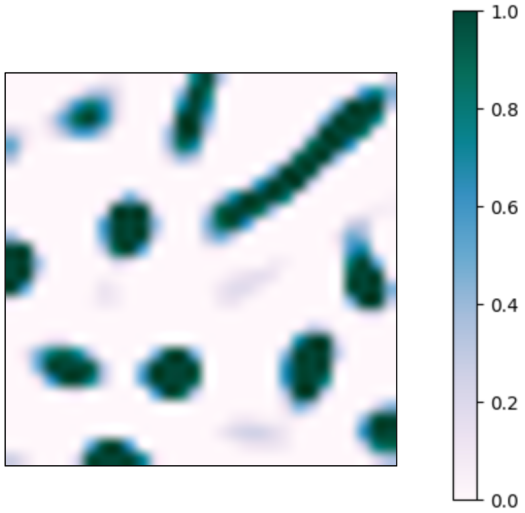

図 6.5: 「2」に分類し直した画像の一例. 斑点のような模様で描画された鉛直風が確認できる.

る．また, 前章で緯度 40$\tilde{6}$0 °, 経度 200 ° 付近の鉛直対流によらない鉛直風の領域を鉛直対流による鉛直風が発生している領域だと識別していたが, 3 グループに分類したことでその領域は正しく鉛直対流がない領域だと識別できていることがわかる．

表 6.4: 3 クラスに分類した場合の予測と正解の比較

予測と正解の比較	予測が 0	予測が 1	予測が 2
正解が 0	930(99.89%)	1(0.11%)	0(0.00%)
正解が 1	86(60.99%)	53(37.59%)	2(1.42%)
正解が 2	10(12.50%)	52(65.00%)	18(22.50%)

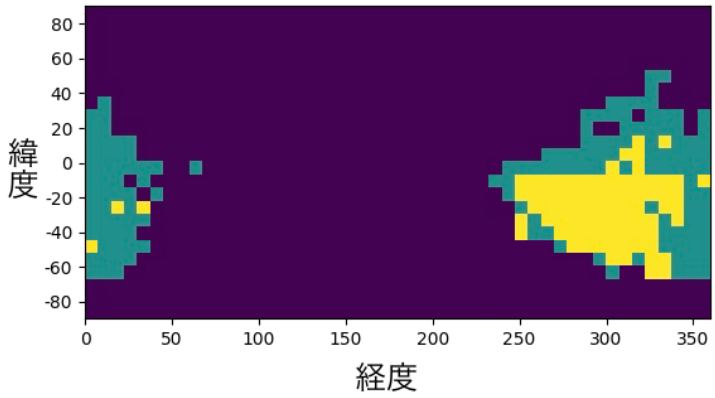

図 6.6: テストデータの正解のラベルを描画した画像. 横軸と縦軸はそれぞれ経度と緯度に沿ったラベルの枚数を示す. 鉛直対流による鉛直風がない領域である「0」のラベルを紫, 筋状の鉛直風が存在する領域である「1」のラベルを緑, 斑点状の鉛直風が存在する領域である「2」を黄色で描画している．

さらに, データの量を 1/2 である 109080 枚に削減し, 同様に学習とモデルによる予測を行った. 予測と正解の比較を表 6.5 に, 予測したラベルを全球で描画した画像を図 6.9 で示している. 2 のラベルの正答率は元のデータ量より高くなっているが, 1 のラベルの正答率は低下している．

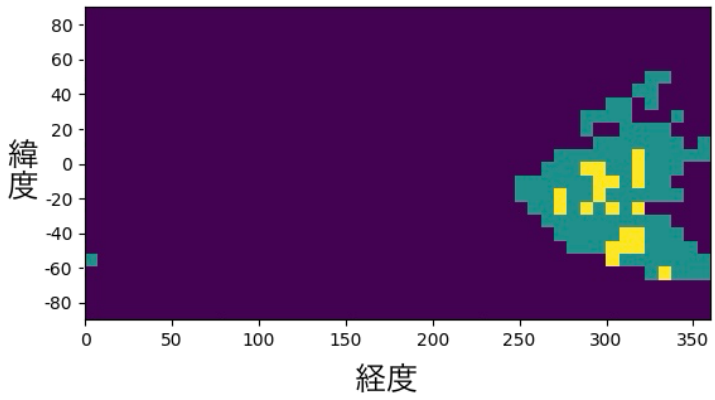

図 6.7: 学習モデルによる予測のラベルを描画した画像. 鉛直対流による鉛直風がない領域である「0」のラベルを紫, 筋状の鉛直風が存在する領域である「1」のラベルを緑, 斑点状の鉛直風が存在する領域である「2」を黄色で描画している.

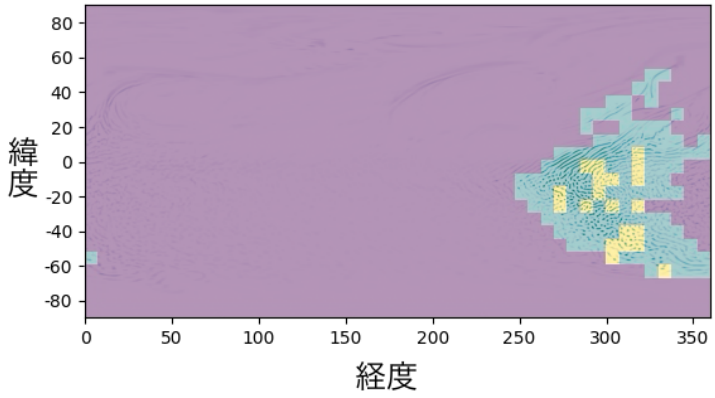

図 6.8: テストデータを描画した画像とモデルによる予測のラベルを重ねた画像. 右側の鉛直風の領域の一部で識別ができていることがわかる.

表 6.5: 半減したデータを使用した場合の予測と正解の比較

予測と正解の比較	予測が 0	予測が 1	予測が 2
正解が 0	930(99.89%)	0(0.00%)	1(0.11%)
正解が 1	79(56.03%)	38(26.95%)	24(17.02%)
正解が 2	8(10.00%)	6(7.50%)	66(82.50%)

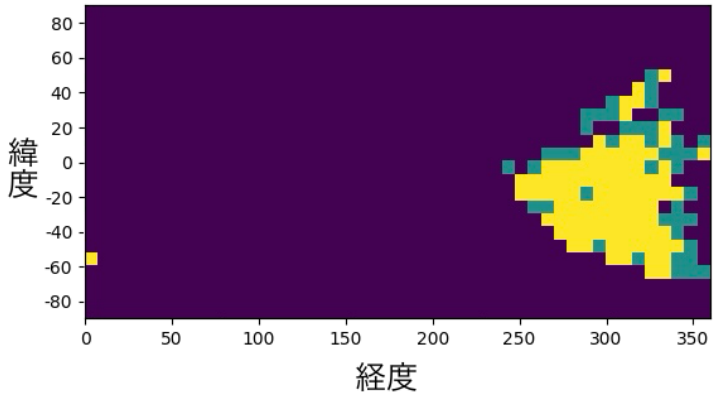

図 6.9: 画像データの枚数を 109080 枚に減らして学習したモデルによる予測のラベルを描画した画像. 鉛直対流による鉛直風がない「0」のラベルを紫, 鉛直対流ではない鉛直風である「1」のラベルを緑, 鉛直対流による鉛直風である「2」を黄色で描画している.

6.3.1 鉛直風がない領域を除いた場合の分類

上記のように, 鉛直風の形状で細分化した 3 グループでの分類を行うと精度が低下することがわかる. そこで, 鉛直風が存在しない領域である「0」のラベルを除き, 図 6.4 で示した筋状の鉛直風と図 6.5 で示した斑点状の鉛直風が存在する領域のデータのみを学習データとして使用し, 計 14157 枚の画像を使用して二値分類モデルの学習を行った. また, 学習モデルによる予測を行い, テストデータに付けた各領域の正解のラベルとの比較を行った. テストデータはこれまでと同様に計

算最終時刻の瞬間場を高度 1125 m で切り出したデータから学習データと同じように鉛直風が存在しない領域を除いた 221 枚のデータを使用した. モデルによる予測と正解の比較を表 6.6 に示す.

表 6.6: 鉛直風がある領域のみで学習を行った場合の予測と正解の比較

予測と正解の比較	予測が 1	予測が 2
正解が 1	134(95.04%)	7(4.96%)
正解が 2	26(32.50%)	54(67.50%)

表 6.6 を見ると, 予測と正解の一致している割合が鉛直風のない領域も含めた 3 グループでの分類の時と比べ高くなっていることが読み取れる. 鉛直風がある領域をより細分化して分類する際は, 鉛直風がない領域のデータを除き学習を行った方が精度が向上すると考えられる.

6.4 結論

本論文での研究の結果, 画像認識技術による膨大な量のデータの処理を実現できた. このような膨大な量のデータの解析は統計的な特徴を捉えることにつながると考えられる. また, 本論文では火星の鉛直対流に着目して研究を行ったが, 今後火星以外の天体や気象現象などにも応用ができる可能性がある.

付録 ソースコード

ここでは本論文で使用したライブラリのバージョンと学習モデルのソースコードを紹介する.
本論文で使用するシミュレーションデータは NetCDF 形式のデータとなっている. NetCDF(Network Common Data Form) は多次元配列を含むことができる自己記述型のデータであり, 拡張子は .nc を使用する. NetCDF 形式のデータは NetCDF4 モジュールを使用することで, Python 上で読み書きが可能となる. Python のバージョンは 3.11.11, netCDF のデータを読み込むために使用した netCDF4 のバージョンは 1.7.2 である.

本論文では CNN モデルの作成と学習のために Python 上でディープラーニングに使用できるライブラリとして tensorflow.keras を使用している. また, Python 上で多次元配列データを扱うための数値計算用のライブラリとして Numpy を使用した. 本研究で使用した際の Numpy のバージョンは 1.26.4 である.
緯度, 経度, 高度, 時間の多次元配列となっている鉛直風のシミュレーションデータは, 分類のために PNG の画像データへと変換している. これは Python のデータ可視化用のライブラリである Matplotlib の中の matplotlib.pyplot モジュールを使用した. 本研究で使用した際の matplotlib.pyplot のバージョンは, 3.10.0 である.
第 2 章で行ったラベル付けでは, 特定の条件を満たすファイルやフォルダのパスを取得できる glob モジュールを使用している. これにより, 分割した各領域のデータを分類したいグループごとのフォルダで分け, glob モジュールを使うことで各領域のデータに対応するグループの 0, あるいは 1 のフォルダ名を取得し, ラベルのデータセットを作成することできる. また, 同様に第 2 章で行ったオーバーサンプリングでは, tensorflow.keras モジュールを使用し画像に回転, 上下反転, 左右反転の処理を加えた.

tensorflow.keras モジュールを使用してニューラルネットワークを構築する場合, 始めに Sequential() コンストラクターでニューラルネットワークの元となる Sequential オブジェクトを生成してから model.add() で畳み込み層などの各層の追加を行う. CNN モデルの畳み込み層は tensorflow.keras の Conv2D() メソッドで設定を行う. filters オプションでフィルタの枚数, kernel_size オプションでフィルタ

のサイズを設定し, $padding = "same"$ と設定を行うことでゼロパディングを行うモデルとなる.$input_shape$ では入力する画像データのサイズを指定することができる .$activation$ では畳み込み層に適用する活性化関数を指定する. 本研究では $ReLU$ 関数を使用したので, $activation = "relu"$ となる.
$Dropout$ ではドロップアウトの割合を指定できる. 本論文のモデルでは 0.5 に設定している.
出力層は $Dense()$ コンストラクターで指定している. 分類したいグループの数に合わせてノード数を決定し, ソフトマックス関数を活性化関数として指定している.
最後に, $compile()$ メソッドで作成した $Sequential$ オブジェクトのコンパイルを行う. この時, 勾配降下法などの重みの最適化に使用する手法 (オプティマイザー) や誤差 (損失関数) の求め方, 学習評価の指標を決定する.

構築した CNN モデルのソースコード

```
# tensorflow.keras モジュールからインポートする
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Dropout,
Flatten
from tensorflow.keras.optimizers import SGD

model = Sequential () # Sequential オブジェクトの生成

# 畳み込み層の設定
model.add(
Conv2D(filters=32, # フィルタの枚数
kernel_size=(3, 3), # フィルタのサイズ
padding="same", # ゼロパディングを行い出力する画像データを同サイズ
にする
input_shape=(30, 30, 1), # 入力する画像データの形状
activation="relu" # 活性化関数は ReLU 関数
))

# Flatten 層の設定
model.add(Flatten())
model.add(Dropout(0.5)) # ドロップアウト

# 出力層の設定
model.add(Dense(2, # 出力層のノード数を決定する
activation="softmax" # 活性化関数はソフトマックス関数
))

model.compile(
loss="sparse_categorical_crossentropy", # 損失関数の設定，正解の
ラベルと予測の間の誤差を求めるスパース行列対応交差エントロピー誤差
optimizer=SGD(learning_rate=0.1), # 重みの最適化を確率的勾配降下
法で行う． 学習率は 0.1
metrics=["accuracy"]) # 学習評価を正解率に設定する

model.summary() モデルの概要を出力
```

本研究で使用した tensorflow.keras モジュールのバージョンは 3.8.0 である．上記のソースコードを実行することで, 構築された CNN のモデルの概要 (表 4.1 参照) が出力される．

次のソースコードは, 学習を実行するためのソースコードである．このソースコードでは早期終了の設定を行うための EarlyStopping クラスを実装している．これは monitor で指定した損失や精度などの監視対象が規定回数改善されなかった場合に, 学習を打ち切るためのクラスである．監視する回数は patience で指定できる．fit() メソッド内の callbacks オプションを使用することで, 1 回の学習が終了するたびに EarlyStopping を呼び出し早期終了の判定を行うことができる．

学習を実行するソースコード

```
%%time
# tensorflow.keras モジュールからインポートする
from tensorflow.keras.callbacks import EarlyStopping
training_epochs = 100 # 学習回数の決定
batch_size = 128 # ミニバッチのサイズ (1 パッチあたりのデータ
# 128 枚 のミニバッチを 926 回，1epoch=926 回
# 148160 枚のうち検証データを除いて 118528 枚を使う

# EarlyStopping(早期終了) 監視対象回数内で改善されなければ学習を止
める
early_stopping = EarlyStopping(
monitor="val_loss", # 監視対象は損失関数
patience=5, # 回数は 5 回 verbose=1 # 早期終了をログとして出力)

history = model.fit(
x_train, # 学習データ (x_train)
y_train, # 学習データの正解ラベル (y_train)
epochs=training_epochs, # 学習を繰り返す回数（エポック）を指定す
る
batch_size=batch_size, # ミニバッチのサイズを指定
verbose=1, # 学習の推移を出力する
validation_split=0.2, # 検証データの割合，20%を使用する
shuffle=True, # 検証データをランダムに抽出する
callbacks=[early_stopping] # 1 エ ポック が 終 了 す る 度 に
EarlyStopping を呼び出す
)

# テストデータを用いて学習したモデルの評価を行う． x_test はテスト
データ，y_test はテストデータの正解ラベル
score = model.evaluate(x_test, y_test, verbose=0)

# テストデータを用いた損失関数を出力
print("Test loss:", score[0])
# テストデータを用いた正解率を出力
print("Test accuracy:", score[1])
```

ソースコードを実行することで, 学習が行われ各学習ごとの損失関数と正解率の推移が出力される.

最後に, モデルによる予測を行うためのソースコードを下に示す. tensorflow.keras に搭載された model.predict() を用いることで, 学習モデルによるテストデータへの予測を行うことができる. 機械学習用のライブラリである scikit-learn に搭載されている sklearn.metrics じゃ学習したモデルの評価を行うためのモジュールである. confusion_matrix メソッドを使用することで, 第 4 章や第 5 章で示した正解と予測の比較の混同行列を出力することができる. pandas はデータ解析用のライブラリであり, ここでは 混同行列を各ラベルの予測に対する正解のラベルの割合として出力するために pandas を使用している.

モデルによる予測を行うソースコード

```
# 必要なライブラリをインポートする
import numpy as np
from sklearn.metrics import confusion_matrix
import pandas as pd

predict = model.predict(x_test, batch_size=16) # テストデータを使用してモデルによる予測を行う
results = predict.argmax(axis=1) # 予測結果の中で最も確率の高いラベルの配列を作成する

# テストデータの正解ラベルとモデルによる予測のラベルの混合行列を出力する
print(confusion_matrix(y_test, results))

pd.options.display.precision = 4 # 表示桁数の設定

# 混合行列を各ラベルの予測に対する正解のラベルの割合として出力する
conf_mat = confusion_matrix(y_test, results, normalize='true')
display(pd.DataFrame(conf_mat))
```

ソースコードを実行することで学習したモデルによる予測が行われ, モデルが予測したラベルとテストデータの正解ラベルの混同行列が出力される. Numpy のバージョンは 1.26.4, scikit-learn モジュールのバージョンは 1.6.1, pandas モジュールは 2.2.2 を使用した.

謝辞

本研究を行うにあたって, 樫村博基講師には研究内容の方針をご指導頂き, 研究や論文執筆に関する相談等で大変お世話になりました. 高橋芳幸准教授には基礎理論読書会やセミナーを通し, 地球流体力学の基礎的な知識や手法についてご指導頂きました. また, 所属する地球および惑星大気科学研究室の皆様には研究室活動を通して発表や論文に関する助言を頂きました. 本論文の執筆に関わってくださった皆様に心からの感謝を申し上げます.

本研究では火星 SCALE-GM で計算を行ったシミュレーションデータを使用しました. 開発, 運用に関わる皆様に深く感謝申し上げます

参考文献

[1] チーム・カルボ. 物体・画像認識と時系列データ処理入門. 株式会社秀和システム, 2021, 584p. 978-4-798-06354-6.

[2] 株式会社アイデミー 山口 達輝, 松田 洋之. 図解即戦力　機械学習&ディープラーニングのしくみと技術がこれ1冊でしっかりわかる教科書. 株式会社技術評論社, 2022, 240p. 978-4-297-10640-9.

[3] 藤野 巖. 入門 ディープラーニング　—NumPy と Keras を使った AI プログラミング—. 株式会社オーム社, 2022, 264p. 978-4-274-22881-0.

[4] 国本大悟, 須藤秋良. スッキリわかる Python 入門 第2版. 株式会社インプレス, 2023, 416p. 978-4-295-01636-6